Ibex PDF Creator

.NET Programmers Guide

For Ibex version 4.7.7.6

Table of Contents

1. Introduction 1

2. Installation 5

3. Getting Started with Ibex 7

4. Introduction to XSL-FO 11

5. Using Ibex 29

6. Using Ibex with ASP.NET 33

7. Error Handling & Logging 39

8. Page Layout 43

9. Text Formatting 53

10. Fonts 65

11. Floats 67

12. Space Handling 69

13. Colors 73

14. Lists 77

15. Tables 81

16. Images 95

17. Scalable Vector Graphics (SVG) images 107

18. Absolute Positioning 123

19. Columns 127

20. Bookmarks 129

21. Configuration 131

22. Extensions 135

23. PDF/X 143

24. Using Ibex with Visual Basic and ASP 147

25. Elements and Attributes 151

Table of Contents

26. Compliance with the XSL-FO standard 373

© 2002-2010 Visual Programming Limited. All rights reserved.

NOTICE: All information contained herein is the property of Visual Programming Limited.

No part of this publication (whether in hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

PostScript is a registered trademark of Adobe Systems Incorporated. Adobe, the Adobe logo, Acrobat, the Acrobat logo, Adobe Garamond, Aldus, Distiller, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photoshop, Poetica, and PostScript are trademarks of Adobe Systems Incorporated. Apple, Mac, Macintosh, QuickDraw, and TrueType are trademarks of Apple Computer, Inc., registered in the United States and other countries. ITC Zapf Dingbats is a registered trademark of International Typeface Corporation. Helvetica and Times are registered trademarks of Linotype-Hell AG and/or its subsidiaries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Times New Roman is a trademark of The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be registered in certain other jurisdictions. Unicode is a registered trademark of Unicode, Inc. All other trademarks are the property of their respective owners.

This publication and the information herein are furnished AS IS, are subject to change without notice, and should not be construed as a commitment by Visual Programming Limited. Visual Programming Limited assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third-party rights.

Chapter 1

Introduction

This manual describes the functionality and use of the Ibex PDF Creator. Ibex is a PDF creation component which can be used from the command line or integrated into a larger application. It ships as a .NET Assembly so it can be used both in stand-alone applications and in server-based applications using ASP and ASP.NET.

This chapter provides an overview of how Ibex works and the process involved in creating PDF files using Ibex.

1.1 The PDF creation process

Ibex creates PDF files which contain data from your application. You provide Ibex with your data in XML format and Ibex creates a PDF file containing that data. The format of the PDF file is specified using an XML template which defines the layout of the document using the elements from the XSL Formatting Objects standard.

The XML you provide to Ibex can come from any source. Typically, it is extracted from a database or generated dynamically for each document.

The formatting objects (FO) standard defines elements such as table, row and cell which can be used to lay out your data on a page. It also defines objects for describing the overall shape and layout of the page, including parameters such as page size, number of columns and regions where content will be placed on the page.

The process of creating a document in FO format is carried out using an XSLT stylesheet. The stylesheet transforms your XML data into standard FO syntax. Ibex then reads that XSL-FO and creates the PDF file. The actual execution of the XSLT translation can be done either by Ibex, which uses the .NET framework XSL translation objects, or externally to Ibex using any XSLT engine.

Figure 1-1 shows some XML data for creating a page which says "Hello world". The corresponding formatting objects from which the PDF is created are shown in Figure 1-2.

Figure 1-1: Simple XML

```
<?xml version="1.0" encoding="UTF-8"?>
<expression>hello world<expression>
```

Figure 1-2: Example formatting objects for hello world

```
<root xmlns="http://www.w3.org/1999/XSL/Format">
  <layout-master-set>
    <simple-page-master master-name="layout" page-width="8.5in" page-height="8in">
      <region-body region-name="body" margin="2.5cm"/>
    </simple-page-master>
  </layout-master-set>
  <page-sequence master-reference="layout">
    <flow flow-name="body">
      <block>Hello world</block>
    </flow>
  </page-sequence>
</root>
```

The process of getting from your data to the formatting objects is carried out using the XSLT stylesheet which you provide.

This approach to PDF creation has a number of advantages:

- the content and presentation are separated. The format of the PDF file is defined by the XSLT stylesheet which can be created and maintained externally to the application. Changing the stylesheet to alter the report layout does not require rebuilding your application;

- formatting elements such as report headers and footers can be maintained in a separate stylesheet which is shared by many reports;

- the formatting objects standard defines a powerful set of objects for creating page content. It supports single-column and multi-column pages, bookmarks, indexing, page headers and footers, complex page numbering, tables with optionally repeating headers and footers and many other features;

- formatting objects is a high-level approach which makes changing report layouts simple. To change the number of columns on a multi-column page you just need to change the column-count attribute on a single element. To do this using a lower level programmatic API takes much more effort.

1.2 Terminology

Ibex uses the FO standard which defines formatting objects such as table and table-cell. FO makes a distinction between two types of objects:

block-level objects broadly speaking these are objects which are formatted vertically down the page. Block level objects are block, block-container, table, table-and-caption, table and list-block.

inline-level objects these are objects whose content appears on lines within a block object. Commonly used inline level objects are external-graphic, inline, leader, page-number, page-number-citation and basic-link.

Other terminology used in XSL-FO includes:

folio number this is the page number which is displayed on a page. This is sometimes different from the physical page number. If you look at this manual in a PDF viewer the physical page number is not the same as the number printed on the page, because some pages such as the front cover are not included in the page number sequence.

1.3 About this manual

This manual is split into two main sections. The first covers an introduction to the use of formatting objects and an overview of various formatting objects such as tables and lists. The second is a reference section listing all the available objects and attributes with many examples of their usage.

This manual was produced with the .NET version of Ibex, release 4.7.7.6.

1.4 About Ibex

Ibex is developed entirely in C# and requires the Microsoft .NET Runtime to be installed. .NET 1.0, 1.1, 2.0, 3.0, 3.5 and 4.0 are supported. All Microsoft operating systems which support .NET are supported by Ibex.

Chapter 2

Installation

The latest version of Ibex can be downloaded from
http://www.xmlpdf.com/downloads-net.html.

The download file is a Windows Installer MSI file which can be installed by double-clicking on it in Windows Explorer.

By default Ibex is installed in the following directory:

c:\program files\visual programming\ibex pdf creator n.n.n

where n.n.n is the version number. So for example Ibex 4.1.2 is installed in:

c:\program files\visual programming\ibex pdf creator 4.1.2

Multiple versions of Ibex can be installed and used on one machine at the same time.

2.1 Assemblies installed - Ibex PDF Creator Standard Edition

The installation process installs these .NET assemblies used by Ibex:

.NET Version	Assembly	Description
1.0 (32bit)	ibex10.dll	Main Ibex assembly
	ibex10.exe	Ibex command line utility
1.1 (32bit)	ibex11.dll	Main Ibex assembly
	ibex11.exe	Ibex command line utility
2.0 (32bit)	ibex20.dll	Main Ibex assembly
	ibex20.exe	Ibex command line utility
3.5 (32bit)	ibex35.dll	Main Ibex assembly
	ibex35.exe	Ibex command line utility
4.0 (32bit)	ibex40.dll	Main Ibex assembly
	ibex40.exe	Ibex command line utility

2.2 Assemblies installed - Ibex PDF Creator Professional Edition

The installation process installs these .NET assemblies used by Ibex:

.NET Version	Assembly	Description
1.0 (32bit)	ibex10.dll	Main Ibex assembly
	ibex10.exe	Ibex command line utility

.NET Version	Assembly	Description
1.1 (32bit)	ibex11.dll	Main Ibex assembly
	ibex11.exe	Ibex command line utility
	ibexshaping11.dll	Provides Arabic text shaping support
2.0 (32bit)	ibex20.dll	Main Ibex assembly
	ibex20.exe	Ibex command line utility
	ibexshaping20.dll	Provides Arabic text shaping support
2.0 (64bit)	ibex20-64.dll	Main Ibex assembly
	ibex20-64.exe	Ibex command line utility
	ibexshaping20-64.dll	Provides Arabic text shaping support
3.5 (32bit)	ibex35.dll	Main Ibex assembly
	ibex35.exe	Ibex command line utility
	ibexshaping35.dll	Provides Arabic text shaping support
3.5 (64bit)	ibex35-64.dll	Main Ibex assembly
	ibex35-64.exe	Ibex command line utility
	ibexshaping35-64.dll	Provides Arabic text shaping support
4.0 (32bit)	ibex40.dll	Main Ibex assembly
	ibex40.exe	Ibex command line utility
	ibexshaping40.dll	Provides Arabic text shaping support
4.0 (64bit)	ibex40-64.dll	Main Ibex assembly
	ibex40-64.exe	Ibex command line utility
	ibexshaping40-64.dll	Provides Arabic text shaping support

2.3 Installing by copying files

Ibex can be installed as part of a larger application by copying the assemblies appropriate to the .NET version being used and registering them in the GAC. It is not necessary to run the Ibex installer to use the Ibex assemblies on test and production machines. We recommend running the installer on development machines as this will install the example files.

2.4 Assemblies not required in production

Typically the command line programs (ibex*.exe) are not used on production servers and do not need to be installed.

The ibexshaping*.dll assemblies are used for shaping Arabic text. These assemblies are loaded dynamically using reflection. If your documents do not contain Arabic text then you do not need to install these assemblies.

Chapter 3

Getting Started with Ibex

Although primarily intended for use as a part of a larger application, the Ibex installation includes command line programs which creates PDF files from XML, XSLT and XSL-FO files. We will use these programs to demonstrate the basics of PDF creation with Ibex.

The command line programs shipped with Ibex are ibex10.exe, ibex11.exe and ibex20.exe. Use the one which corresponds to the version of .NET you have installed.

Throughout this manual an XML file which uses the XSL formatting objects vocabulary is referred to as an FO file or just the FO.

The command line syntax for all versions is the same. In these examples we use ibex20.exe.

3.1 Ibex command line program usage

To create a PDF file from a FO file specify the names of the FO and PDF files on the command line. For example to create hello.pdf from hello.fo you do this:

```
ibex20 hello.fo hello.pdf
```

If the names of the input and output files are the same (ignoring the extensions) you can abbreviate this to:

```
ibex20 hello.fo
```

and if the file extension of the input file is "fo" or "xml" you can abbreviate even further to:

```
ibex20 hello
```

3.2 Error logging

Any informational or error messages will be logged to the console. To send any error messages to a file as well, use the -logfile option. For example, to log errors to the file ibex.log the command becomes:

```
ibex20 -logfile ibex.log hello.fo hello.pdf
```

3.3 An example without XSLT translation

The Ibex command line program will create a PDF file from either (a) an FO file or (b) an XML file with an XSLT stylesheet. This section shows how to create a PDF file from an FO file.

This example uses the FO file hello.fo shown in Figure 3-1.

Figure 3-1: Hello World FO

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.w3.org/1999/XSL/Format">

  <layout-master-set>
    <simple-page-master master-name="page">
      <region-body margin="2.5cm" region-name="body"/>
    </simple-page-master>
  </layout-master-set>

  <page-sequence master-reference="page">
    <flow flow-name="body">
     <block>Hello World</block>
    </flow>
  </page-sequence>
</root>
```

Each of the elements and attributes used in the file is explained later in the manual. For now we just want to get started with using the Ibex command line program.

Using the command

```
ibex20 hello
```

creates the file hello.pdf containing the text "Hello World".

3.4 An example with XSLT translation

The Ibex command line program will create a PDF file from either (a) an FO file or (b) an XML file with an XSLT stylesheet. This section shows how to create a PDF file from an XML data file with an XSLT stylesheet.

Using Ibex without having Ibex do the XSLT transformation to create the FO is useful if you have created the FO using another tool or if you just want to manually change some FO to experiment with layout.

In practice XSLT is almost always part of the PDF creation process because XSL-FO lacks some simple features such as being able to sequentially number headings. The designers of XSL-FO presumed that XSLT would be used and so did not duplicate features already in XSLT.

Ibex gives you the flexibility of having Ibex do the XSLT translation or having some other tool do it. Internally Ibex uses the XSLT translation classes provided by .NET. The XSLT classes in .NET 2.0 are significantly faster and use memory more efficiently than in .NET 1.0 and 1.1. For this reason we recommend using .NET 2.0 or later.

In this example we will translate some XML with an XSLT stylesheet and produce a PDF from the result of the translation.

We have some weather forecast data in the file weather.xml. This file contains the XML shown in Figure 3-2.

Figure 3-2: Weather Forecast Data

```
<?xml version="1.0" encoding="UTF-8"?>
<forecast>
   <city name="Wellington" temp="20"/>
</forecast>
```

We also have an XSLT stylesheet weather.xsl which contains the XSL shown in Figure 3-3.

Figure 3-3: Weather Forecast Stylesheet

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format"
   xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<xsl:strip-space elements="*"/>

<xsl:template match="forecast">

  <root xmlns="http://www.w3.org/1999/XSL/Format">

    <layout-master-set>
      <simple-page-master master-name="page-layout">
         <region-body margin="2.5cm" region-name="body"/>
      </simple-page-master>
    </layout-master-set>

    <page-sequence master-reference="page-layout">
       <flow flow-name="body">
           <xsl:apply-templates select="city"/>
       </flow>
    </page-sequence>

  </root>

</xsl:template>

<xsl:template match="city">
   <fo:block>
      <xsl:value-of select="@name"/>

      <xsl:value-of select="@temp"/>
   </fo:block>
</xsl:template>

</xsl:stylesheet>
```

This template outputs the root, layout-master-set and page-sequence elements. Then for each city record in the data outputs a block element using the template shown in Figure 3-4.

Figure 3-4: weather-data-xsl-subset

```
<xsl:template match="city">
   <block>
      <xsl:value-of select="@name"/>
       
      <xsl:value-of select="@temp"/>
   </block>
</xsl:template>
```

We can translate and format this example using the command:

```
ibex20 -xsl weather.xsl weather.xml weather.pdf
```

The result of this translation is the file weather.pdf

3.5 Required skills

To use Ibex you need know how to edit XSL stylesheets. Some familiarity with XSLT is required, although in depth knowledge is not. The Ibex website contains examples of using XSLT for common document related functions such as creating a table of contents.

Familiarity with XSL-FO is not required. This manual contains enough information to enable you to produce complex documents using Ibex.

Chapter 4

Introduction to XSL-FO

This chapter provides an overview of formatting objects and provides some suggestions on how to create PDF documents from XML files. We also look at the techniques for using XSLT transformation to create FO files.

4.1 Layout of an FO file

A very simple FO file is shown in Figure 4-1:

Figure 4-1: Simple FO file

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">

   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </simple-page-master>
   </layout-master-set>

   <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>

</root>
```

This file is logically in three parts, namely the root, layout-master-set and page-sequence parts. All FO files share this structure.

4.1.1 Namespaces

The examples used in this manual follow the style shown in Figure 4-1, where the XSL-FO namespace is set (on the root element) as the default namespace for the file. Namespace prefixes are not used for the FO elements such as block. Figure 4-2 shows the same FO as Figure 4-1 but without the default namespace. Each element has the "fo:" namespace prefix. The files shown in Figure 4-1 and Figure 4-2 both create the same output and are treated equally by Ibex. Using namespaces is a matter of preference, it does not effect performance.

Figure 4-2: Simple XML using the fo prefix

```
<?xml version='1.0' encoding='UTF-8'?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <fo:layout-master-set>
  <fo:simple-page-master master-name="simple">
      <fo:region-body margin="2.5cm" region-name="body"
        background-color='#eeeeee'/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="simple">
    <fo:flow flow-name="body">
      <fo:block>Hello World</block>
    </fo:flow>
  </fo:page-sequence>

</fo:root>
```

4.1.2 The root element

The root element shown in Figure 4-3 contains the whole content of the file and establishes the XSL-FO namespace as the default namespace. This element is the same for all FO files.

Figure 4-3: The root element

```
<root xmlns="http://www.w3.org/1999/XSL/Format">
```

Additional namespaces can be added to the xml element as shown in Figure 4-4.

Figure 4-4: The root element with additional namespaces

```
<root xmlns="http://www.w3.org/1999/XSL/Format"
            xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format"
            xmlns:svg="xmlns="http://www.w3.org/2000/svg"
>
```

4.1.3 The layout-master-set element

The layout-master-set element show in Figure 4-5 defines the shape and layout of pages in the document. Within the layout-master-set we have a simple-page-master element which in turn contains the region-body element.

The simple-page-master defines the layout of one type of page and is uniquely identified by its master-name attribute. The region-body element defines an area of the page where content will be placed. A page can have more than one region so we give the region a unique name "body" using the region-name attribute. This value is used with flow elements to specify which content goes into which region on the page.

Figure 4-5: The master-layout element

```
<layout-master-set>
   <simple-page-master master-name="simple">
      <region-body margin="2.5cm" region-name="body"
         background-color="#eeeeee"/>
   </simple-page-master>
</layout-master-set>
```

A FO file contains one or more simple-page-master elements, each with a unique master-name. In this simple example we have only one. Each simple-page-master element creates a formatting object known as a *page master*.

An example of a more complex document is the Ibex manual. Each chapter begins with a page which has no header. This is followed by a page which has left-aligned footer, then a page with a right-aligned footer. Each of the three possible page layouts is defined by a different simple-page-master element.

4.1.4 The page-sequence element

The page-sequence element shown in Figure 4-6 defines a sequence of pages that will appear in the PDF document. The master-reference attribute is used to tie the content of the page-sequence to a particular page layout, in this case one defined previously using a simple-page-master. When Ibex finds a page-sequence element it looks at the list of known simple-page-master and page-sequence-master elements (we have no page-sequence-master elements in this example) and finds one with a master-name attribute which equals the master-reference attribute on the page-sequence. If Ibex does not find a matching page master the FO file is invalid and Ibex will throw an exception.

Figure 4-6: The page-sequence element

```
<page-sequence master-reference="simple">
  <flow flow-name="body">
     <block>Hello World</block>
  </flow>
</page-sequence>
```

Within the page-sequence element we have a flow element. This holds the content which will appear on one or more pages. A page can have multiple regions. To associate content with a region we use the flow-name attribute on the flow element. In order for the content contained in the flow to appear on the page, the flow-name of the flow should match a region-name of one of the regions (in this example the region-body) on the page.

If the flow-name of the flow does not match a region-name of one of the regions on the page the content is not displayed on that page. This is not an error. It is a useful feature and we show how to use it later in this chapter.

Looking at the FO in Figure 4-7, the underlined names must match each other, and the names in *italics* should match if you want the content to appear.

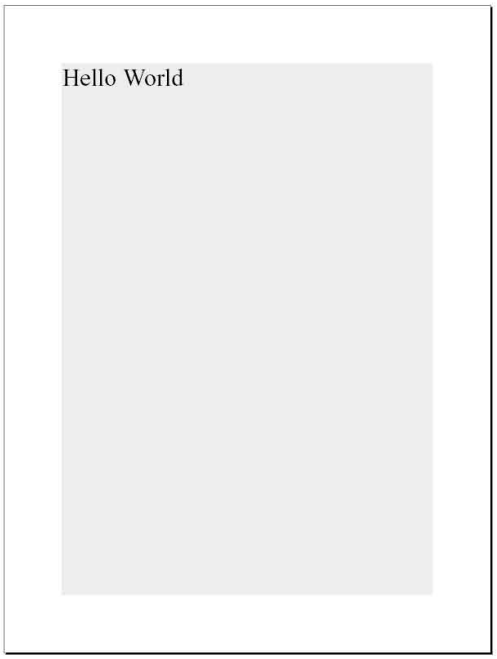
Figure 4-7: Matching master-name and master-reference

```
<layout-master-set>
  <simple-page-master master-name="simple">
     <region-body margin="2.5cm" region-name="body"
        background-color='#eeeeee'/>
  </simple-page-master>
</layout-master-set>

   <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
```

Within the flow element we can have one or more "block level" elements. These are elements such as list, block and table which define the content to appear on the page. In this example we have a single block element containing the text "Hello World".

This produces a page like the one shown in Figure 4-8. The region created by the region-body element has a shaded background so you can see how large it is.

Figure 4-8: A basic page with a region-body and some text



4.2 Adding a footer region

In our example so far all the text contained in the flow element goes into the body region in the center of the page. To add a page footer we need to define a new region on the page and then define some new content to go into that region.

We define a footer region by adding a region-after element into the existing simple-page-master as shown in Figure 4-9.

Figure 4-9: Simple page master with footer region

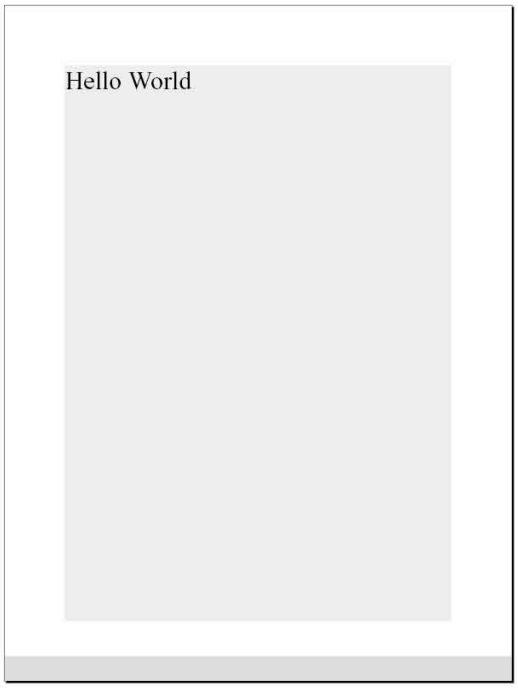
```
<layout-master-set>
   <simple-page-master master-name="simple">
     ...
      <region-after extent='1cm' region-name="footer"
        background-color='#dddddd'/>
     ...
   </simple-page-master>
</layout-master-set>
```

The region-after element defines an area on the page which extends the full width of the page. If we had side regions (region-start and region-end) this might change, but in this example we have no side regions.

The height of the region created by the region-after element is defined by the extent attribute. In this example we have extent="1cm", so the region will be 1cm high and end at the bottom of the page.

Even without any content the footer region is still rendered on the page. Our page now looks like the one in Figure 4-10.

Figure 4-10: A basic page with a footer region



In its current position on the page the footer region will not print on most printers because they do not print right to the edge of the paper. We can define a margin around the whole page by setting the margin attribute on the simple-page-master element of the page-sequence as shown in Figure 4-11.

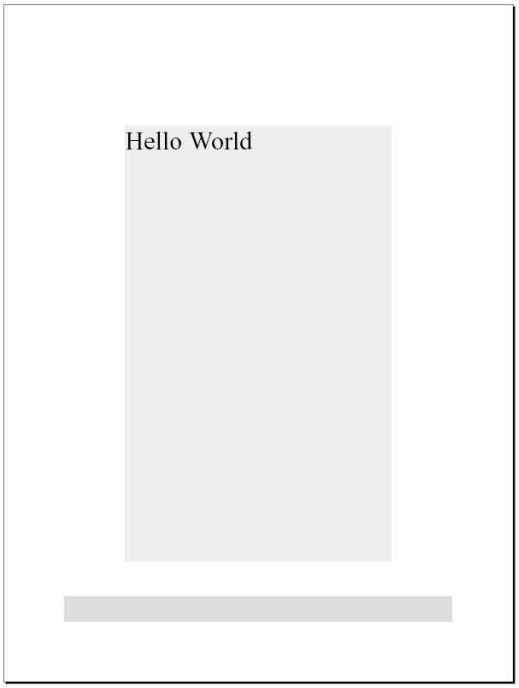
Figure 4-11: Simple page master with margin added

```
<layout-master-set>
  <simple-page-master master-name="simple"
          margin="2.5cm">
     <region-body margin="2.5cm" region-name="body"
        background-color="#eeeeee"/>
     <region-after extent="1cm" region-name="footer"
        background-color="#dddddd"/>
  </simple-page-master>
</layout-master-set>
```

The area inside the margins of the simple-page-master is called the "content area". The area covered by the regions (defined by the region-body and region-end) is measured from the inside of the page's content area, so when we add margins to the simple-page-master we reduce the size of the regions correspondingly.

Our page now appears as shown in Figure 4-12.

Figure 4-12: After adding margins to the simple-page-master



Now that we have some space on the sides of the body region we can remove the side margins from the body by changing the definition from that shown in Figure 4-13 to the one shown in Figure 4-14, resulting in the page layout shown in Figure 4-15.

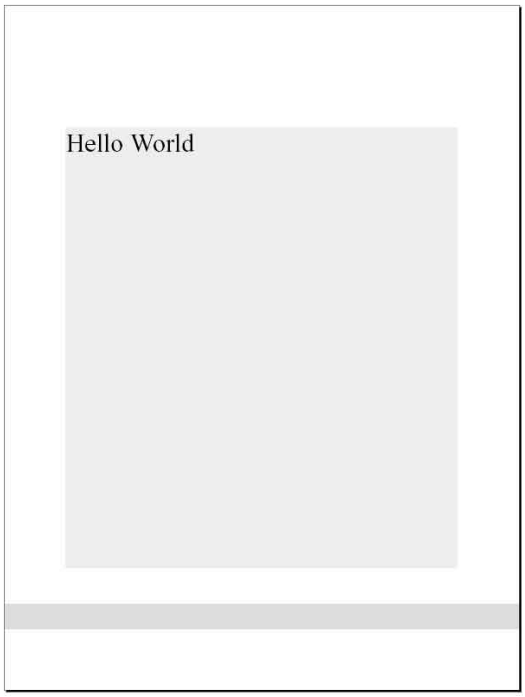
Figure 4-13: Body with side margins

```
<region-body margin="2.5cm" region-name="body"
     background-color="#eeeeee"/>
```

Figure 4-14: Body without side margins

```
<region-body margin-top="2.5cm" margin-bottom="2.5cm"
   region-name="body"  background-color="#eeeeee"/>
```

Figure 4-15: After removing the left and right margins from the region-body



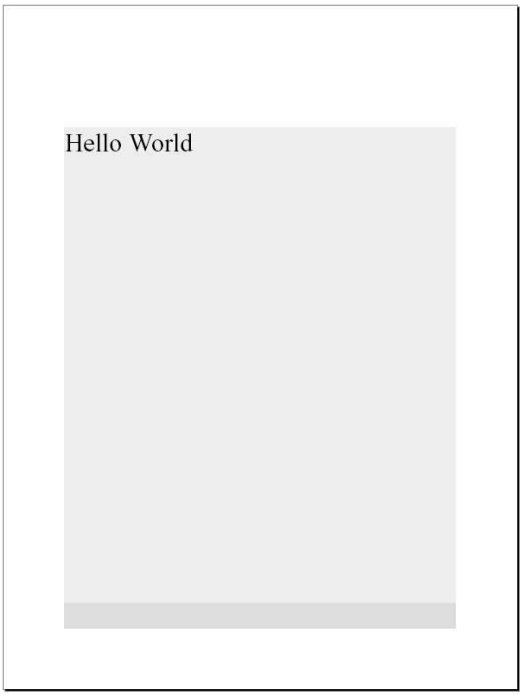
The last thing we need to do to get a working page layout is to make the footer region narrower by adding side regions. The left side region is created with a region-start element and the right side with a region-end element as in Figure 4-16. We can also specify the bottom-margin attribute of the body region so that it ends just where the footer starts, by setting margin-bottom= "1cm" on the region-body element.

Figure 4-16: side regions

```
<layout-master-set>
  <simple-page-master master-name="simple"
        margin='2.5cm'>
     <region-body margin="2.5cm" margin-bottom='1cm'
         region-name="body"
        background-color='#eeeeee'/>
     <region-after extent='1cm' region-name="footer"
        background-color='#dddddd'/>
     <region-start extent='2.5cm'/>
     <region-end extent='2.5cm'/>
  </simple-page-master>
</layout-master-set>
```

By default the side regions take precedence over the top and bottom regions so the top and bottom regions become narrower. This gives us the page layout shown in Figure 4-17, to which we can start adding some content.

Figure 4-17: With side regions to reduce the width of the footer



4.3 Attribute processing

The FO above also illustrates one of the ways in which XSL-FO handles attributes. We can specify a shorthand attribute such as "margin", which has the effect of setting the specific values margin-left, margin-right, margin-top and margin-bottom, and then override just the specific value we want (by setting margin-bottom="1cm"). The order in which the attributes are specified has no effect. A more specific setting will always override a more general one. So the two examples in Figure 4-18 and Figure 4-19 produce the same result.

Figure 4-18: Shorthand and specific attributes

```
<layout-master-set>
  <simple-page-master master-name="simple">
     <region-body margin="2.5cm" margin-bottom="1cm">
  </simple-page-master>
</layout-master-set>
```

Figure 4-19: Shorthand and specific attributes

```
<layout-master-set>
  <simple-page-master master-name="simple">
     <region-body margin-bottom="1cm" margin="2.5cm">
  </simple-page-master>
</layout-master-set>
```

4.4 Adding content to the footer

While content is added to the body of the page using the flow element, content is added to other regions using the static-content element. The "static" part of the static-content name refers to the fact that the content defined in this element stays within the region specified on this page. It does not flow from one page to the next. If the content exceeds the size of the region it will not flow to the next page.

The content of the static-content is repeated on every page which has a region with a matching flow-name (such as "footer"), and is typically different on every page as the page number changes.

To insert a simple footer with the words "XSL-FO Example" we add a static-content element as shown in Figure 4-20.

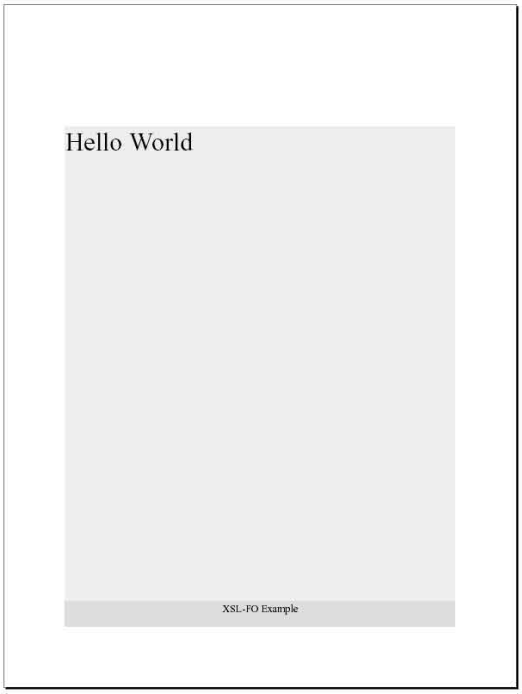
Figure 4-20: Adding a static-content element

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple"
           margin='2.5cm'>
         <region-body margin="2.5cm" margin-bottom='1cm'
            region-name="body"  background-color='#eeeeee'/>
         <region-after extent='1cm' region-name="footer"
            background-color='#dddddd'/>
         <region-start extent='2.5cm'/>
         <region-end extent='2.5cm'/>
      </simple-page-master>
   </layout-master-set>
   <page-sequence master-reference="simple">
      <static-content flow-name="footer">
          <block text-align='center'>
          XSL-FO Example</block>
      </static-content>
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
</root>
```

Note that the order of the static-content and flow elements is important. All static-content elements must come before any flow elements.

This FO produces the page shown in Figure 4-21.

Figure 4-21: FO with static-content



Note that the flow-name of the static-content element and the region-name of the region-after element must match for the content to appear. This feature makes it

possible to have many static-content elements within the same page-sequence, and only those which match regions in the current simple-page-master will be rendered.

The Ibex manual has three different page layouts defined with three different simple-page-master elements. Each simple-page-master has a footer region with a different region-name. The main flow element contains three different static-content elements all containing footers. Only the footer whose flow-name matches the region-name for the currently active simple-page-master will be rendered.

4.5 Adding the page number to the footer

To insert the current page number into the document use the page-number element inside the static-content element as shown in Figure 4-22.

Figure 4-22: Adding a page number

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple"
           margin='2.5cm'>
         <region-body margin="2.5cm" margin-bottom='1cm'
            region-name="body"  background-color='#eeeeee'/>
         <region-after extent='1cm' region-name="footer"
            background-color='#dddddd'/>
         <region-start extent='2.5cm'/>
         <region-end extent='2.5cm'/>
      </simple-page-master>
   </layout-master-set>
   <page-sequence master-reference="simple">
      <static-content flow-name="footer">
          <block text-align='center'>
          XSL-FO Example, page <page-number/>
          </block>
      </static-content>
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
</root>
```

This FO produces the page shown in Figure 4-23.

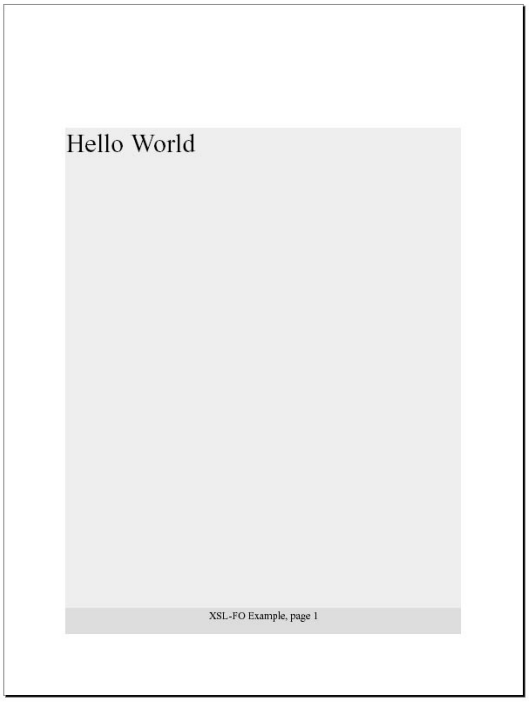


Figure 4-23: Page with page number

4.6 Adding the total page count to the footer

Adding the total page count (so we can have "page 3 of 5") is a two step process, based on the use of the "id" attribute which uniquely identifies an FO element. We place a block on the last page with the id of "last-page", and then we use the page-number-citation element to get *the number of the page on which that block appears* as our total number of pages.

Typically the block with the id of "last-page" is empty so a new page is not created at the end of the document.

The FO for the last block in the document is shown in Figure 4-24, and the FO to retrieve the last page number and put it in the footer is shown in Figure 4-25.

Figure 4-24: Block with id for last page

```
<block id="last-page"/>
```

Figure 4-25: FO to retrieve the page number of the identified block

```
<page-number-citation ref-id="last-page"/>
```

You can see how the id and ref-id values match. This is how Ibex associates the two elements and knows from which block to retrieve the page number.

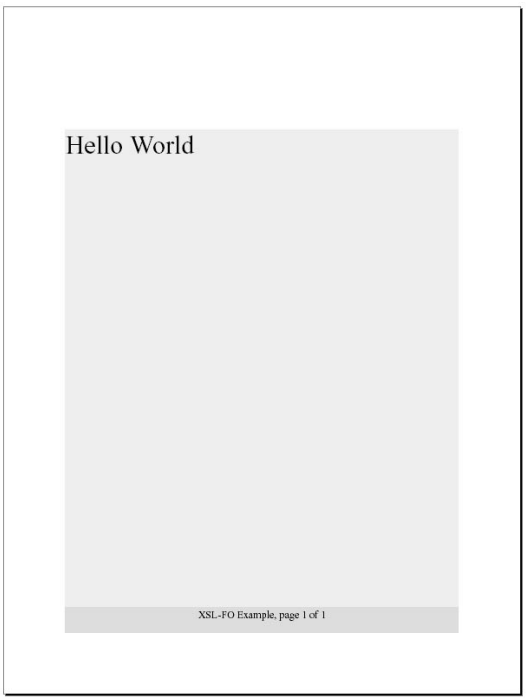
So bringing all these elements together we have the FO shown in Figure 4-26.

Figure 4-26: Complete FO to display total page count

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple"
            margin='2.5cm'>
         <region-body margin="2.5cm" margin-bottom='1cm'
            region-name="body"  background-color='#eeeeee'/>
         <region-after extent='1cm' region-name="footer"
            background-color='#dddddd'/>
         <region-start extent='2.5cm'/>
         <region-end extent='2.5cm'/>
      </simple-page-master>
   </layout-master-set>
   <page-sequence master-reference="simple">
      <static-content flow-name="footer">
          <block text-align='center'>
          XSL-FO Example, page <page-number/>
          of <page-number-citation ref-id='last-page'/>
          </block>
      </static-content>
      <flow flow-name="body">
         <block>Hello World</block>
         <block id='last-page'/>
      </flow>
   </page-sequence>
</root>
```

This FO produces the page shown in Figure 4-27.

Figure 4-27: Page with page number and total page count



4.7 Adding text content

Text is added to the body region of the page by using the block element. A block element can contain any amount of text and has attributes which define how the text will appear. These attributes are described in more detail later in the manual.

A block can contain text as shown in Figure 4-28.

Figure 4-28: Text in a block

```
<flow flow-name="body">
   <block>Hello World</block>
</flow>
```

A block element can also contain other block elements which in turn contain text or more nested elements. Figure 4-29 shows a block which contains another block with a different font, set using the font attribute.

Figure 4-29: Nested blocks

```
<flow flow-name="body">
     <block>
          Hello World
          <block font-size="16pt">
             this is a nested block
          </block>
     </block>
  </flow>
```

There is no limit to the nesting of block elements.

4.8 Using borders and padding

Many FO elements can have a border around the area they create on the page. If the border around an element is the same on all four sides then it can be defined with a single use of the border attribute. The space between a border and the content of the block (in this case the text) is controlled using the padding attribute. Figure 4-30 shows FO for a block with border and padding.

Figure 4-30: Block with border and padding

```
<flow flow-name="body">
  <block background-color='#eeeeee'>
    <block>
         Hello World
    </block>
    <block border='1pt solid red' padding='3pt'>
         Hello World
    </block>
  </block>
</flow>
```

This example has two block elements nested inside an outer element, with a background color set on the outer element to highlight the area created by the outer block. The block created from this FO is shown in Figure 4-31.

Figure 4-31: Default indentation of nested blocks

```
Hello World
Hello World
```

Ibex positions the content of the block (in this case the text) relative to the edge of the region. After positioning the content, the padding and borders are positioned relative to the position of the content. This places the padding and borders *outside* the content area of the block. The contents of the block are not indented, rather the padding and borders extend outside the block. This is the default behavior of XSL-FO formatters.

If you prefer Cascading Style Sheets (CSS) compatible behavior where adding a border to a block indents its content, you can specify the left-margin and right-margin attributes to force this to happen. Even if the left-margin and right-margin values are zero, CSS type indentation will still occur. The XML for this is shown in Figure 4-32 and the resulting output is shown in Figure 4-33.

Figure 4-32: Block with margins specified

```
<flow flow-name="body">
  <block background-color='#eeeeee'>
    <block>
      Hello World
    </block>
    <block border='1pt solid red' padding='3pt' margin-left="0" margin-right="0">
      Hello World
    </block>
  </block>
</flow>
```

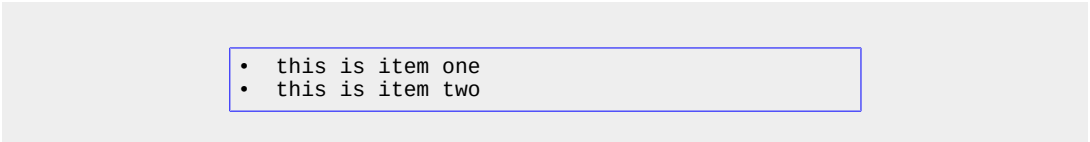
Figure 4-33: Default indentation of nested blocks

4.9 Creating lists

A list is content divided into two columns. Each item in the list is in two parts, called the *label* and the *body* respectively. A list is created with the list-block element. A list-block contains one or more list-item elements, each of which contains exactly one list-item-label element and one list-item-body element.

An example of a simple list is shown in Figure 4-34.

Figure 4-34: Example of list-block



This list was created with the FO shown in Figure 4-35:

Figure 4-35: FO for the list-block

```
<list-block
   margin-left='3cm' margin-right='3cm' padding='3pt'
   border='.1pt solid blue'
   provisional-distance-between-starts='0.5cm'
   provisional-label-separation='0.1cm'>
   <list-item>
       <list-item-label end-indent='label-end()'>
           <block>&#x2022;</block>
       </list-item-label>
       <list-item-body start-indent='body-start()'>
           <block>this is item one</block>
       </list-item-body>
   </list-item>

   <list-item>
       <list-item-label end-indent='label-end()'>
          <block>&#x2022;</block>
       </list-item-label>
       <list-item-body start-indent='body-start()'>
          <block>this is item two</block>
       </list-item-body>
   </list-item>
</list-block>
```

A list sets the two columns to widths specified using the attributes of the list-block elements. The provisional-distance-between-starts attribute specifies the distance between the start of the label column and the start of the body column. The provisional-label-separation attribute sets how much of the label column should be left empty to provide a blank space between the columns.

If we expand the above example and add more content to the first body we can see that the content is constrained to the column. If we add content to the first list-item-body as shown in Figure 4-36 we get the list shown in Figure 4-37.

Figure 4-36: FO for the list-block

```
<list-item-body start-indent='body-start()'>
  <block>
     If your Network Administrator has enabled it,
     Microsoft Windows can examine your network and
     automatically discover network connection settings.
  </block>
</list-item-body>
```

Figure 4-37: Output for the list-block

- If your Network Administrator has enabled it, Microsoft Windows can examine your network and automatically discover network connection settings.
- this is item two

For more information on lists see page 77.

4.10 Creating tables

A table is created using the table element. Within the table element there can be one table-header element, any number of table-body elements (which contain the rows) and one table-footer element. Each of the table-header, table-body and table-footer

elements contains one or more table-row elements, each containing one or more table-cell elements which in turn contain block-level elements such as block, table and list-block.

Since a table-cell can contain any block-level element you can easily create tables which contain text, nested tables or lists. Table headers and footers defined with the table-header and table-footer elements are automatically repeated at each page break, although this can be suppressed if required.

Figure 4-38 shows the FO for a simple table and Figure 4-39 shows the table created from the FO.

Figure 4-38: FO for a table

```
<table>
  <table-body>
    <table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 2</block>
       </table-cell>
    </table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 2</block>
       </table-cell>
    </table-row>
  </table-body>
</table>
```

This FO produces the table shown in Figure 4-39.

Figure 4-39: Simple table from the above FO

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

The padding and border attributes are not inherited from containing elements so must be defined on the table-cell elements.

4.10.1 Setting table column widths

The width of a table column is set using the table-column element. A table element contains zero or more table-column elements each of which defines properties such as width and background-color for a column in the table.

To make the first column 30% of the table width we would add table-column elements as shown in Figure 4-40, which creates the output shown in Figure 4-41.

Figure 4-40: Table with table-column elements

```
<table>

   <table-column column-width='30%' column-number='1'/>
   <table-column column-width='70%' column-number='2'/>

  <table-body>
    <table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 2</block>
       </table-cell>
    </table-row>
    <table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
          <block>row 1 column 2</block>
       </table-cell>
    </table-row>
  </table-body>
</table>
```

Figure 4-41: Rendered table with specified widths

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

For more information on tables see page 81.

Chapter 5

Using Ibex

This chapter describes how to call the Ibex API and how to use the accompanying command line program.

5.1 Ibex command line program

Although primarily intended to be used as a part of a larger application, Ibex ships with a command line program which can be used to create PDF files from FO files.

The command line programs shipped with Ibex are ibex10.exe (which uses .NET 1.0), ibex11.exe (which uses .NET 1.1) and ibex20.exe (which uses .NET 2.0).

The command line syntax is the same for all programs. In these examples we use ibex20.exe.

To create a PDF file from an FO file specify the file names on the command line. For instance to create hello.pdf from hello.fo, you do this:

```
ibex20 hello.fo hello.pdf
```

5.1.1 XSLT translation

The command line program will accept XML data and an XSLT stylesheet as inputs. The XML will be translated to FO by the stylesheet and the results then formatted to PDF. The command line syntax is:

```
ibex20 -xsl xsl-file xml-file pdf-file
```

So to create a PDF file from the files book.xml and book.xsl, the command is:

```
ibex20 -xsl book.xsl book.xml book.pdf
```

XSLT parameters can be passed to the stylesheet by adding them as name-value pairs to the command line. For instance, if we want to define the parameter called "age" to the value "30" we use a command like this:

```
ibex20 -xsl book.xsl book.xml hello.pdf "age=30"
```

The use of the double quotes around the name-value pair is necessary on some operating systems to force them to come through as a single parameter to the Ibex program.

5.1.2 Logging from the command line

Any informational or error messages will be logged to the console. To send error messages to a file as well, use the -logfile option. For example to log errors to the file ibex.log, you would do this:

```
ibex20 -logfile ibex.log hello.fo hello.pdf
```

5.1.3 Listing available fonts

You can also list the fonts which are available (based on what fonts are installed on your system) by using the -fonts option like this:

```
ibex20 -fonts
```

The list of fonts is produced as a FO file to the standard output. This can be redirected to a file and then used as input to Ibex to create a PDF file containing a table which looks like this:

	file	usage	example
minion	c:\windows\fonts\MOB_____.TTF	10pt minion	**10pt minion**
	c:\windows\fonts\MOB_____.TTF	bold 10pt minion	**bold 10pt minion**
	c:\windows\fonts\MOI_____.TTF	italic 10pt minion	*italic 10pt minion*
	c:\windows\fonts\MOBI____.TTF	bold italic 10pt minion	***bold italic 10pt minion***

The list of fonts can be limited to fonts whose name contains a specified string by passing the string on the command line. For instance if we wanted to see what versions of "arial" are installed, we can use the command:

```
ibex20 -fonts arial
```

5.2 The Ibex API

A PDF document is generated using the FODocument object which is in the ibex4 namespace.

First you create a new FODocument object and then calling the generate() method on that object. The generate() method has various versions which take different parameters depending on whether the input is from files or streams and whether XSLT translation should occur.

The FODocument object is not thread safe. A new FODcoument should be created for each PDF file to be created. Ibex does support creating multiple PDF files concurrently on multiple threads, as long as each PDF file is associated with a unique FODocument instance.

Example C# code to convert the file "manual.fo" to "manual.pdf" the code is shown in Figure 5-1, the equivalent VB.NET code is in Figure 5-2.

Figure 5-1: C# code to create a PDF from an FO file

```
using System;
using ibex4;

public class Simple {

   static void Main( string[] args ) {

      FODocument doc = new FODocument();

      gen.generate( "manual.fo", "manual.pdf" );
   }

}
```

Figure 5-2: VB.NET code to create a PDF from an FO file

```
Imports System
Imports ibex4

Module Module1

    Sub Main()

        Dim doc As New FODocument

        doc.generate("manual.fo", "manual.pdf")

    End Sub

End Module
```

Projects need to have a reference to the ibex DLL.

5.2.1 Generating to File

```
public void generate( string fo_file_name, string pdf_file_name )
```

This will read the FO contained in the file named in pdf_file_name and create the PDF file named in pdf_file_name.

5.2.2 Generating using streams

```
public void generate( Stream fo_stream, Stream pdf_stream )

public void generate( Stream fo_stream, Stream pdf_stream, bool close_stream )
```

This will read the FO from the System.IO.Stream called fo_stream and create the PDF file into the System.IO.Stream pdf_stream. These streams can be anything derived from System.IO.Stream, such as System.IO.FileStream or System.IO.MemoryStream.

If close_stream is true the PDF stream will be closed after the PDF file is generated, if false it will not. By default the stream is closed. Not closing the stream is useful if you are generating to a MemoryStream object as the bytes cannot be read from the MemoryStream if it has been closed.

5.2.3 Generating a PDF from XML and XSL

These methods take XML, an XSLT stylesheet, and a stream to write the resulting PDF file to.

```
public void generate( Stream xml_stream, Stream xsl_stream, Stream pdf_stream )

public void generate( Stream xml_stream, Stream xsl_stream, Stream pdf_stream, bool
closeStream )
```

Ibex uses the .NET XSLT processor to transform the XML using the specified stylesheet and passes the resulting FO to the PDF creation routines. XSLT transformation is faster or more efficient in .NET 2.0 and later and we recommend using this version or later if possible.

5.2.4 Generate a PDF from XML and XSL with parameters

These methods are similar to the ones in the previous section but take an additional hashtable which (if not null) should contain name-value pairs which are then passed as arguments to the XSLT translation process.

```
public void generate( Stream xml_stream, Stream xsl_stream, Stream pdf_stream,
                      bool close_stream, Hashtable params )
```

Chapter 6

Using Ibex with ASP.NET

This chapter shows how to use Ibex with ASP.NET, including how to create a PDF file and stream it back to a client browser without needing to save it to disk.

Ibex creates a PDF file into a Stream object (from the System.IO namespace). In this example we use a MemoryStream object which derives from Stream and is basically an expandable array of bytes held in memory. Creating the PDF file in a MemoryStream means the file is not saved to disk and the whole process occurs in memory.

We create the file into a MemoryStream because many versions of Internet Explorer will display a PDF file correctly only if they know how long the file is. We need to create the PDF file into a MemoryStream to determine its length and set this length in an HTTP header. This need to know the length of the PDF file prevents us from streaming directly to the Response object.

6.1 The ASP page

This example uses a page called pdfasp.aspx, which has code behind it in pdfasp.aspx.cs. To use these pages you will need to create a new ASP.NET C# project, add a reference to the ibex20.dll wherever you have installed it, and then add the pdfgen.aspx page. You will need to remove the '_' characters from the names of the pdfasp.aspx and pdfasp.aspx.cs files.

The content of the ASP page pdfasp.aspx is shown in Figure 6-1.

Figure 6-1: The ASP page

```
<%@ Page language="c#" validateRequest="false" Codebehind="pdfasp.aspx.cs"
  AutoEventWireup="false" Inherits="WebApplication1.pdfasp" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>pdfgen</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
        </form>
    </body>
</HTML>
```

6.2 The ASP code-behind page

The pdfasp.aspx.cs file contains the code shown in Figure 6-2 (plus some error handling that is not shown for clarity).

Figure 6-2: The code-behind page

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Text;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Xml.Xsl;
using System.Security.Policy;

using ibex4;
using ibex4.logging;

namespace WebApplication1 {

  public class pdfasp : System.Web.UI.Page {

  private void Page_Load(object sender, System.EventArgs e) {

     Logger.getLogger()
        .setLevel( Level.FINEST )
        .clearHandlers();

     try {
       Stream stream = new FileStream(
              Server.MapPath( @"logs\log.txt"), FileMode.Append,
                  FileAccess.Write ) ;

       Logger.getLogger().addHandler( new StreamHandler( stream ) ) ;
      }
      catch( Exception ) {
      }

      FODocument doc = new FODocument();

      string dataFilePath = Server.MapPath("") + "\\hello.fo";

      // stream for output in memory before sending to browser
      MemoryStream pdfStream = new MemoryStream();

      FileStream dataStream = new FileStream( dataFilePath,
             FileMode.Open, FileAccess.Read );

      Logger.getLogger().info( "data file path is " + dataFilePath );

      using( dataStream ) {
          doc.generate( dataStream, pdfStream, false );
      }

      Response.Clear();

      Response.ContentType = "application/pdf";
      Response.AddHeader( "content-length",
            System.Convert.ToString( pdfStream.Length ) );

      Response.BinaryWrite( pdfStream.ToArray() );
      Response.End();
   }

   override protected void OnInit(EventArgs e)   {
      InitializeComponent();
      base.OnInit(e);
```

```
  }

  private void InitializeComponent() {
     this.Load += new System.EventHandler(this.Page_Load);
  }
 }
}
```

Key points about the process of streaming the PDF are covered in the following sections.

6.2.1 Logging

We set up logging so if an error occurs we get the error logged to file:

```
Logger.getLogger()
    .setLevel( Level.FINEST )
    .clearHandlers();

try {
    Stream stream = new FileStream( Server.MapPath( @"logs\log.txt"),
           FileMode.Append, FileAccess.Write ) ;

    Logger.getLogger().addHandler( new StreamHandler( stream ) ) ;
}
catch( Exception ) {
}
```

6.2.2 Setting the MIME type

We set the correct MIME type to cause the browser to invoke the Acrobat plugin to display the PDF:

```
Response.ContentType = "application/pdf";
```

6.2.3 Setting the content length

We set the content length to IE will correctly handle the content:

```
Response.AddHeader( "content-length",
        System.Convert.ToString( pdfStream.Length ) );
```

6.2.4 Not closing the MemoryStream

We call the version of generate() that takes a boolean to indicate we do not want the output stream to be closed. If it were closed we would not be able to extract the PDF data from the MemoryStream in order to copy it back to the browser. Typically if we

were writing the PDF to a file on disk we would close the stream but in this case we need to stream to stay open.

```
using( dataStream ) {
     doc.generate( dataStream, pdfStream, false );
}
```

6.3 Using XSLT

If we want to translate our XML data using a stylesheet the ASP code needs to be changed as shown in Figure 6-3.

Figure 6-3: ASP page using XSLT

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Text;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Xml.Xsl;
using System.Security.Policy;

using ibex4;
using ibex4.logging;

namespace WebApplication1 {

  public class pdfasp : System.Web.UI.Page {

  private void Page_Load(object sender, System.EventArgs e) {

     Logger.getLogger().setLevel( Level.FINEST ).clearHandlers();

     try {
       Stream stream = new FileStream(
              Server.MapPath( @"logs\log.txt"), FileMode.Append,
                  FileAccess.Write ) ;

       Logger.getLogger().addHandler( new StreamHandler( stream ) ) ;
      }
      catch( Exception ) {
      }

      FODocument doc = new FODocument();

      string dataFilePath = Server.MapPath("") + "\\data.xml";
      string templateFilePath = Server.MapPath("") + "\\template.xsl";

      // stream for output in memory before sending to browser
      MemoryStream pdfStream = new MemoryStream();

      FileStream dataStream = new FileStream( dataFilePath,
             FileMode.Open, FileAccess.Read );

      FileStream xslStream = new FileStream( templateFilePath,
             FileMode.Open, FileAccess.Read );


      Logger.getLogger().info( "data file path is " + dataFilePath );

      using( dataStream ) {
          using( xslStream ) {
```

```
            doc.generate( dataStream, xslStream, pdfStream, false );
      }
      }

      Response.Clear();

      Response.ContentType = "application/pdf";
      Response.AddHeader( "content-length",
            System.Convert.ToString( pdfStream.Length ) );

      Response.BinaryWrite( pdfStream.ToArray() );
      Response.End();
   }

   override protected void OnInit(EventArgs e)   {
      InitializeComponent();
      base.OnInit(e);
   }

   private void InitializeComponent() {
      this.Load += new System.EventHandler(this.Page_Load);
   }
 }
}
```

The changes made were:

- we declared a variable for the XSLT template like this:

```
string templateFilePath = Server.MapPath("") + "\\template.xsl";
```

- we opened the XSLT template using a Stream

```
FileStream xslStream = new FileStream( templateFilePath, FileMode.Open,
FileAccess.Read );
```

- we passed the XSL stream to the generate() method:

```
using( dataStream ) {
  using( xslStream ) {
    doc.generate( dataStream, xslStream, pdfStream, false );
  }
}
```

Chapter 7

Error Handling & Logging

This chapter describes error handling using the Ibex API.

Ibex associates an error handler with the library as a whole. Generally this error handler will log a message and not throw an exception.

The Ibex Logger object is a singleton which is retrieved using a call to the ibex4.logging.Logger.getLogger() method. Typically you would import the ibex4.logging namespace and then access the logger as shown in Figure 7-1.

Figure 7-1: Clearing existing error handlers

```
using ibex4.logging;

void sumfunc() {
  Logger.getLogger().clearHandlers();
}
```

The default error handler writes messages to the console. Messages are displayed in various circumstances including:

- when an invalid attribute is found;
- when a reference is made to a font or image file which cannot be found;
- when a formatting error occurs, such as defining the widths of columns in table that exceed the available width.

As the Ibex Logger is a singleton object, **logging should be configured once at the start of an application, not on a per-document basis.**

7.1 Error severity

To change the level of information logged you can set the level on the logging object to one of the values defined in the ibex4.logging.Level object. Possible levels of logging which can be set are:

SEVERE WARNING INFO CONFIG FINE FINER FINEST

An example of how to set the logger to log only messages which are WARNING or worse is shown in Figure 7-2.

Figure 7-2: Setting the error level

```
using System;

using ibex4;
using ibex4.logging;

public class Create {

  public static void Main( string[] args ) {

    PDFDocument doc = new PDFDocument();

    Logger.getLogger().setLevel( Level.WARNING );
```

7.2 Logging to a file

To log messages to a file, create an ibex4.logging.FileHandler object and then tell the logger to log to this object. The example in Figure 7-3 logs to the file "log.txt", but any valid file name can be used.

Figure 7-3: Logging to a file

```
using System;
using ibex4;
using ibex4.logging;

public class Create {

  public static void Main( string[] args ) {

    Logger.getLogger()
      .setLevel( Level.SEVERE )
      .clearHandlers()
      .addHandler(
        new FileHandler("log.txt") );
```

The FileHandler object synchronises access to the log file.

If you omit the clearHandlers() call shown in the above example, log records will be written to the default console handler and also to the file handler. You will see error messages on the console and they will also be written to the file.

7.3 Logging to a stream

Ibex can log messages to a stream created by the caller. The stream is any object which implements the System.IO.Stream interface.

To log messages to a stream, create an ibex4.logging.StreamHandler object and then tell the logger to log to this object. The example in Figure 7-4 logs to a MemoryStream, but any valid stream can be used.

Figure 7-4: Logging to a stream

```
using System;
using System.IO;
using ibex4;
using ibex4.logging;

public class Create {

  public static void Main( string[] args ) {

    Logger.getLogger().clearHandlers();
    MemoryStream stream = new MemoryStream();
    StreamHandler h = new StreamHandler( stream );
    Logger.getLogger().addHandler( h )
```

If you omit the clearHandlers() call shown in the above example log records will be written to the default console handler and to the stream handler as well.

7.4 Logging to multiple destinations

Errors can be logged to any number of handlers. The example in Figure 7-5 logs to a file called "xslfo.log", to a memory stream and to the console.

Figure 7-5: Logging to multiple destinations

```
using System;
using System.IO;

using ibex4;
using ibex4.logging;

public class Create {

public static void Main( string[] args ) {

   MemoryStream stream = new MemoryStream();

   Logger.getLogger()
        .addHandler( new ConsoleHandler() )
        .addHandler( new StreamHandler( stream ) )
        .addHandler( new FileHandler("xslfo.log") );
   }
}
```

Chapter 8

Page Layout

This chapter describes how to configure the size of a page and position the regions in which content appears.

8.1 Using one layout for all pages

The first element in any FO file is the root element which contains the whole FO tree defining the document and declares the XML namespaces used. Figure 8-1 shows a simple FO file.

Figure 8-1: Simple FO file

```
<root xmlns="http://www.w3.org/1999/XSL/Format">
  <layout-master-set>
    <simple-page-master master-name="layout" page-width="8.5in" page-height="8in">
      <region-body region-name="body" margin="2.5cm"/>
    </simple-page-master>
  </layout-master-set>
  <page-sequence master-reference="layout">
    <flow flow-name="body">
      <block>Hello world</block>
    </flow>
  </page-sequence>
</root>
```

The first FO element within the root element is the layout-master-set element. This contains one or more simple-page-master elements which define the layout of a page, including the width and height.

The simple-page-master element is like a template for a page, defining the page size and the areas on the page into which content will be placed. As content is read from a flow, Ibex decides which simple-page-master to use as the basis for creating the current page. If there is only one simple-page-master then it is always used. If there are several simple-page-masters then a selection process is used to see which one applies to the current page.

The simple-page-master element contains region elements such as region-body which define an area on the page which can be filled with text or image content.

There can be any number of simple-page-master elements provided each has a unique master-name attribute.

Figure 8-2 shows an example of a layout-master-set.

Figure 8-2: Example layout-master-set

```
<layout-master-set>
  <simple-page-master master-name="front-page">
```

```
    <region-body margin-right="2.5cm"
           margin-left="4cm"
           margin-bottom="4cm"
           margin-top="4cm" region-name="body"
     background-color="#eeeeee"/>
    <region-after extent="3cm" region-name="footer"
       background-color='#dddddd'/>
  </simple-page-master>
</layout-master-set>
```

This shows a layout-master-set which contains a single simple-page-master with a master-name of "front-page".

This simple-page-master defines a page which has two regions on which content can be printed. A page defined with this layout appears in the examples at the end of this chapter, on page 49. For the purposes of this example the regions have background-colors defined to show them clearly. More complex layouts showing five regions appear in the examples on page 49.

Having defined a page layout which has a name, (defined by its master-name attribute) we then use the page-sequence element to define the content of the document. The page-sequence element has a master-name attribute which should match the master-name defined for a simple-page-master (or a page-sequence-master, more of which later).

A page-sequence for printing "Hello World" is shown in Figure 8-3.

Figure 8-3: page-sequence for hello world

```
<page-sequence master-reference="front-page">
  <flow flow-name="body">
    <block>Hello World</block>
  </flow>
</page-sequence>
```

A key thing to note is that the content of the page-sequence is contained in a flow element. For content of the flow to appear on the PDF page *the flow-name attribute of the flow element must match the region-name of a region on the page master specified by the master-reference on the page-sequence.* If the flow-name does not match a region-name, none of the content of this flow will appear in the output document.

It is important to understand this feature. It means that a page-sequence can contain multiple flow and static-content elements each containing a flow element with a different flow-name. Only flow elements whose flow-name attribute matches a region-name defined in the current page sequence will appear. This is how we produce different formats for odd and even pages.

Figure 8-4 shows in matching colors the attributes which should match for content to appear.

Figure 8-4: Matching flow and region names

```
<root xmlns="http://www.w3.org/1999/XSL/Format">
  <layout-master-set>
    <simple-page-master master-name="front-page">
      <region-body margin-right="2.5cm"
           margin-left="4cm"
           margin-bottom="4cm"
           margin-top="4cm" region-name="body"/>
      <region-after extent="3cm" region-name="footer"/>
    </simple-page-master>
  </layout-master-set>

  <page-sequence master-reference="front-page">
    <flow flow-name="body">
```

```
      <block>Hello World</block>
    </flow>
  </page-sequence>
</root>
```

8.2 Using different layouts for different pages

It is possible to define different page layouts for different pages. This can be done in two possible ways, either by assigning different page masters to different page sequences, or by using a page-master-alternatives element which chooses from a set of simple-page-master elements based on criteria such as the current page number.

8.2.1 Using different page masters for each page sequence

Using a different page master for each page sequence is useful when you can clearly divide the document into distinct sections. For example, this manual has a different page master for the front cover and for the pages in the table of contents. The page masters for this are shown in Figure 8-5.

Figure 8-5: Two page masters

```
<layout-master-set>

  <simple-page-master master-name="front-page" margin="1.5cm" page-height="297mm"
page-width="210mm">
    <region-body region-name="body" margin="0.75cm 0.5cm 0.75cm 3cm"/>
    <region-before region-name="header" extent="2.5cm"/>
    <region-after region-name="footer" extent="1cm"/>
    <region-start extent="1cm" background-color="#eeeeee"/>
  </simple-page-master>


  <simple-page-master master-name="toc-page" margin="1.5cm" >
    <region-body column-count="1" region-name="body" margin="0.75cm 0.5cm 1cm 3cm"
margin-left="2cm" margin-right="1.5cm" />
    <region-before region-name="header" extent="1cm"/>
    <region-after region-name="footer" extent="0.75cm"/>
    <region-start extent="2cm" />
    <region-end region-name="end" extent="1.5cm" />
  </simple-page-master>

</layout-master-set>
```

Content is allocated to the two sections of the document using two separate page-sequences, as shown in Figure 8-6.

Figure 8-6: Allocating content to two page masters

```
<page-sequence master-reference="front-page">
   <flow flow-name="body">
      <block>
        content that appears in the body of the front page
      </block>
  </flow>
</page-sequence>

<page-sequence master-reference="toc-page">
   <flow flow-name="body">
      <block>
        content that appears in the table of contents
      </block>
  </flow>
</page-sequence>
```

When using this approach content from one flow always appears on pages with the same layout. Flowing content across different page layouts is described in the next section.

8.2.2 Using page master alternatives

Often it is desirable to have content flow continuously across pages with different layouts. This is done in the Ibex manual, where the pages are laid out like this:

first page of chapter	has no page header page number is on the right of the footer
even numbered page	has a page header page number is on the left of the footer
odd numbered page	has a page header page number is on the right of the footer

The three page masters are shown in Figure 8-7.

Figure 8-7: Page masters for three different layouts

```
<simple-page-master  master-name="chapter-odd-no-header">
  <region-body region-name="body" margin="2.5cm 2.5cm 2.5cm 4.0cm"/>
  <region-after region-name="footer-odd" extent="1.5cm" display-align="before"/>
</simple-page-master>

<simple-page-master master-name="chapter-even">
  <region-body region-name="body" margin="2.5cm 2.5cm 2.5cm 4.0cm" column-count="1"/>
  <region-before region-name="header-even" extent="1.5cm" display-align="after"/>
  <region-after region-name="footer-even" extent="1.5cm" display-align="before"/>
</simple-page-master>

<simple-page-master  master-name="chapter-odd">
  <region-body region-name="body" margin="2.5cm 2.5cm 2.5cm 4.0cm"/>
  <region-before region-name="header-odd" extent="1.5cm" display-align="after"/>
  <region-after region-name="footer-odd" extent="1.5cm" display-align="before"/>
</simple-page-master>
```

To make content from a single flow element span multiple pages with different page layouts we use a page-sequence-master element as shown in Figure 8-8. This element contains a repeatable-page-master-alternatives element, which in turn contains a set of conditional-page-master-reference elements.

When formatting content from a page-sequence which has flow-name="chapter", Ibex looks at each of the conditional-page-master-reference elements and chooses which one will be active for the current page. This is done by evaluating conditions specified with the page-position attribute. As a page is created, each conditional-page-master-reference is considered in turn, starting from the first one. The first one found whose conditions are satisfied will determine the page master for the

current page. **Since alternatives are considered in the order in which they appear in the FO, the order in which the alternatives are listed is important.**

When the first page of the chapter is being created, the page-position="first" condition is true, so the first conditional-page-master-reference will be chosen because it has page-position = "first". This has master-reference = "chapter-odd-no-header", so the simple-page-master with master-name = "chapter-odd-no-header" becomes the active page master for the first page of the chapter.

When the second page of the chapter is being created, the page-position="first" is no longer true so the conditions on the next conditional-page-master-reference will be evaluated.

Although not shown in this example, other attributes such as blank-or-not-blank can be used to control the selection of one of the alternatives.

Figure 8-8: The page-sequence-master element

```
<page-sequence-master master-name="chapter" >
  <repeatable-page-master-alternatives>

    <conditional-page-master-reference page-position="first"
             master-reference="chapter-odd-no-header"/>

    <conditional-page-master-reference odd-or-even="odd"
             master-reference="chapter-odd"/>

    <conditional-page-master-reference odd-or-even="even"
             master-reference="chapter-even"/>

  </repeatable-page-master-alternatives>
</page-sequence-master>
```

region-before

region-body

This page layout is created with the XML below. Note that by default the region-start and region-end regions extend the full height of the page and the region-before and region-after regions are narrowed so as not to overlap the side regions. See the following page for an example where the precedence attribute is used to change this.

```
<simple-page-master master-name="region-example-1">

    <region-body margin="2.5cm" region-name="body"
       background-color="#eeeeee"/>

    <region-before extent="2.5cm" region-name="header"
       background-color="#dddddd"/>

    <region-after extent="2.5cm" region-name="footer"
       background-color="#dddddd"/>

    <region-start extent="2.5cm" region-name="start"
       background-color="#aaaaaa"/>

    <region-end extent="2.5cm" region-name="end"
       background-color="#aaaaaa"/>

</simple-page-master>
```

region-after

region-before region-example-1-margins

region-body

This page layout is created with the XML below. Note that by default the region-start and region-end regions extend the full height of the page and the region-before and region-after regions are narrowed so as not to overlap the side regions. See the following page for an example where the precedence attribute is used to change this.
This layout differs from the previous page in that the simple-page-master has the margin attribute set to "2.5cm". This creates a margin of 2.5cm around the entire page, and regions are positioned with respect to the rectangle created by the margins, not with respect to the edges of the paper.

```
<simple-page-master
master-name="region-example-1M" margin="2.5cm">

    <region-body margin="2.5cm"
region-name="body"
       background-color="#eeeeee"/>

    <region-before extent="2.5cm"
region-name="header"
       background-color="#dddddd"/>

    <region-after extent="2.5cm"
region-name="footer"
       background-color="#dddddd"/>

    <region-start extent="2.5cm"
region-name="start"
       background-color="#aaaaaa"/>

    <region-end extent="2.5cm"
region-name="end"
       background-color="#aaaaaa"/>

</simple-page-master>
```

region-after

region-before

region-body

region-after

This page layout is created with the XML below. Note that the region-before and region-after regions have precedence="true" so they extend the full width of the page and the side regions are reduced in height to the regions do not overlap.

```
<simple-page-master master-name="region-example-1">

    <region-body margin="2.5cm" region-name="body"
       background-color="#eeeeee"/>

    <region-before extent="2.5cm" region-name="header"
       precedence="true" background-color="#dddddd"/>

    <region-after extent="2.5cm" region-name="footer"
       precedence="true" background-color="#dddddd"/>

    <region-start extent="2.5cm" region-name="start"
       background-color="#aaaaaa"/>

    <region-end extent="2.5cm" region-name="end"
       background-color="#aaaaaa"/>

</simple-page-master>
```

Chapter 9

Text Formatting

Text is created in the output document using the block element.

The simplest possible block is shown in Figure 9-1.

Figure 9-1: A simple block

```
<block>hello world</block>
```

This creates a paragraph in the output document which has the default font (which is helvetica) and the default alignment (which is left).

The sections below describe elements and attributes used to control the formatting of text.

9.1 Using the font attribute

The quickest way to get the font you require is to use the `font` attribute, as shown in Figure 9-2.

Figure 9-2: Using the font attribute

```
<block font="bold 12pt garamond">hello world</block>
```

Using the font attribute is simpler than specifying all the individual attributes such as font-weight and font-size, but does need some care. When using the font attribute the order of the words is important. The font style (normal, italic) and the font weight (bold, normal) must come before the font size. The font name must come after the font size. If the font name contains spaces, it must be enclosed in quotes, as shown in Figure 9-3.

Figure 9-3: A font name with spaces

```
<block font="bold 12pt "times new roman"">
      hello world
</block>
```

The full syntax of the font attribute is shown in Figure 9-4.

Figure 9-4: Syntax of font attribute

```
[ [ <font-style> || <font-variant> || <font-weight> ]?
<font-size> [ / <lineheight>]?
<font-family> ]
```

9.2 Using the font-family attribute

The font-family attribute is used to specify the name of the font to use. More than one font name can be listed. These names can be specific font names such as "times roman" or "garamond", or generic names such as "monospace". Ibex will use the first name in the list which matches a font on your system. Font names are separated by a comma.

The ability to list multiple font names derives from the CSS standard. It is designed to support the creation of a web page which will be rendered on a computer that may not have the same fonts installed as the page's author. In practice when you generate a PDF file you know what fonts you have installed, so you will probably just specify one font.

9.3 Italic text

Text is made italic using the font-style attribute.

The font style can be "normal" or "italic". Other font values such as the font-family are inherited from the current font, as shown in Figure 9-5. The output created by the FO in Figure 9-5 is shown in Figure 9-6.

Figure 9-5: Using font-style

```
<block font-family="arial">
  hello <inline font-style="italic">world</inline>
</block>
```

Figure 9-6: Using the font-style attribute

hello *world*

9.4 Bold text

Text is made bold using the font-weight attribute.

The font weight can be "normal" or "bold", as shown in Figure 9-7. The output created by the FO in Figure 9-7 is shown in Figure 9-8.

Figure 9-7: Using the font-weight attribute

```
<block font-family="arial">
  hello <inline font-weight="bold">world</inline>
</block>
```

Figure 9-8: Using font-weight

hello **world**

9.5 Text size

The size of text is set using the font-size attribute.

The font size specifies the size of the font and can be specified in a number of ways listed below.

A numeric size

The most common approach is to specify the size you want in points, for example font-size="12pt" or font-size="30pt".

An absolute size

Attribute Value	Size
xx-small	7.0pt
x-small	8.3pt
small	10.0pt
medium	12.0pt
large	14.4pt
x-large	17.4pt
xx-large	20.7pt

A relative size

This sets the font size based on the size of the prevailing font.

Attribute Value	Size
smaller	existing size / 1.2
larger	existing size * 1.2

Another way of setting the font size relative to the current font size is to use the "em" unit. "1.0em" is the current font size, so "2.0em" specifies a size which is twice as big as the current size.

9.6 Underlining text

Text is underlined using the text-decoration attribute.

Specifying text-decoration="underline" will cause text to be underlined, like this.

9.7 Striking out text

You can strike out text using the text-decoration attribute.

Specifying text-decoration="line-through" will cause text to be underlined, like ~~this~~.

9.8 Horizontal alignment

Horizontal alignment is specified using the text-align attribute. The default alignment is left.

Valid values for text-align are shown in the table below.

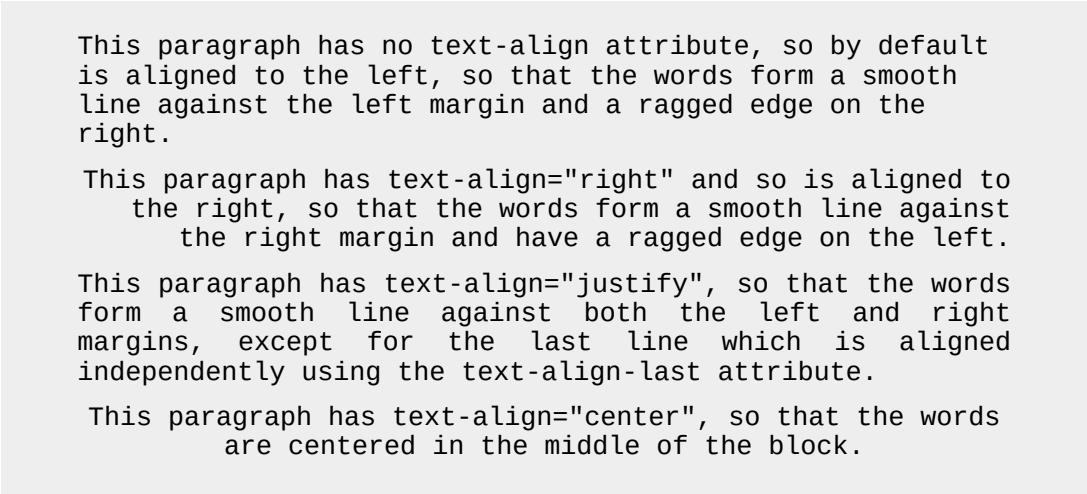
Value	Effect
left	text is aligned against the left edge of the block
right	text is aligned against the right edge of the block

Value	Effect
center	text is centered in the middle of the block
justify	text is aligned against both the left and right edges of the block. Space is inserted between words to achieve this effect. Setting text-align = "justify" does not align the last line of the paragraph, this is done using text-align-last = "justify".
start	text is aligned against the start edge, which for a block that is not rotated, with the default left-to-right writing direction, is the left edge.
end	text is aligned against the end edge, which for a block that is not rotated, with the default left-to-right writing direction, is the right edge.
inside	assuming the document is to be bound as a book, text is aligned against the edge which is nearest the binding. For an odd-numbered page this will be the left edge, for an even numbered page it will be the right edge.
outside	assuming the document is to be bound as a book, text is aligned against the edge which is furtherest from the binding. For an odd-numbered page this will be the right edge, for an even numbered page it will be the left edge.

For text-align values of "inside" and "outside" the page number is used to determine the binding edge, which is assumed to be the left hand edge of odd-numbered pages and the right hand edge of even-numbered pages.

The effect of some of the text-align values is shown in Figure 9-9.

Figure 9-9: Effects of text-align values


This paragraph has no text-align attribute, so by default is aligned to the left, so that the words form a smooth line against the left margin and a ragged edge on the right.
This paragraph has text-align="right" and so is aligned to the right, so that the words form a smooth line against the right margin and have a ragged edge on the left.
This paragraph has text-align="justify", so that the words form a smooth line against both the left and right margins, except for the last line which is aligned independently using the text-align-last attribute.
This paragraph has text-align="center", so that the words are centered in the middle of the block.

9.8.1 Justifying the last line of a paragraph

Specifying text-align="justify" will justify all lines of a paragraph except the last. This is because a justified paragraph typically looks like the one in Figure 9-10, with the last line not being justified.

Figure 9-10: Paragraph without the last line justified

```
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc mollis, turpis
vehicula aliquam auctor, metus turpis tempus justo, eu gravida nisl nibh vitae nisl.
Cras a nisl. Integer et metus vitae dui placerat egestas. Duis rutrum. Nulla in
enim. Suspendisse vel massa in mauris sagittis pharetra. Etiam hendrerit euismod
velit. Ut laoreet lectus nec nisl.
```

The text-align-last attribute controls the alignment of the last line of a paragraph. Values include are shown in the table below:

Value	Effect
relative	if text-align is "justify", align the last line against the start edge (normally the left edge), otherwise use the setting if the text-align attribute.
left	text is aligned against the left edge of the block
right	text is aligned against the right edge of the block
start	text is aligned against the start edge, which for a block that is not rotated, with the default left-to-right writing direction, is the left edge.
end	text is aligned against the end edge, which for a block that is not rotated, with the default left-to-right writing direction, is the right edge.
inside	assuming the document is to be bound as a book, text is aligned against the edge which is nearest the binding. For an odd-numbered page this will be the left edge, for an even numbered page it will be the right edge.
outside	assuming the document is to be bound as a book, text is aligned against the edge which is furtherest from the binding. For an odd-numbered page this will be the right edge, for an even numbered page it will be the left edge.
justify	justify the last line across the whole width of page.

9.9 Left and right margins

The margins of a block are specified using the margin-left and margin-right attributes.

The margin properties indent the edge of the paragraph by the specified amount from the edge of the containing area.

The FO for a block with a 2.5cm left margin is shown in Figure 9-11.

Figure 9-11: Setting the left margin

```
<block margin-left="2.5cm">hello world</block>
```

If we nest another block inside this one, as shown in Figure 9-12, the margins are cumulative. The output from this FO is shown in Figure 9-13.

Figure 9-12: Nested blocks

```
<block margin-left="2.5cm">
  block 1
  <block margin-left="2.5cm">
    block 2
  </block>
</block>
```

Figure 9-13: Output from the above FO

```
                block 1
                                block 2
```

Putting background colors on the blocks shows this more clearly. The FO is in Figure 9-14 and the output is in Figure 9-15.

Figure 9-14: Nested blocks with background color

```
<block margin-left="2.5cm" background-color="#777777">
  block 1
  <block margin-left="2.5cm"  background-color="#999999">
    block 2
  </block>
</block>
```

Figure 9-15: Output from above FO

```
                block 1
                                block 2
```

The approach to indentation defined in the XSL-FO standard is that *the content* of two nested blocks *which do not specify a margin* have the same left edge. The edges of the content (which in our example is the text) are aligned, and any borders and padding are placed outside those edges. Figure 9-16 shows the FO for two nested blocks with no margin attributes. The text will be vertically aligned and the background colors will be placed outside the text. Figure 9-17 shows the resulting output.

Figure 9-16: Nested blocks with no margins specified

```
<block padding="1cm" background-color="#777777">
  block 1
  <block padding="1cm" background-color="#999999">
    block 2
  </block>
</block>
```

Figure 9-17: Output from nested blocks with no margins

```
block 1
block 2
```

In XSL-FO terms, both areas have the same start-indent and hence the same content rectangle, and the padding on the outer block extends outside its content rectangle. This may seem counter-intuitive to some developers used to the CSS model. You can invoke the CSS nested areas model by specifying a margin-left value, even "0pt".

9.10 Spacing between letters

The amount of space between two letters is dependent on the font used. Ibex reads the TrueType or Type 1 font file and loads the width of each character. Kerning information which specifies adjustments to the gaps between particular pairs of characters is also read from the font file and used in the text formatting process.

The spacing between letters can be changed using the letter-spacing attribute. Any value specified using this attribute is *added* to the default spacing specified by the font.

Figure 9-18 shows the FO to increase the letter spacing of some text. The resulting text is shown in Figure 9-19.

Figure 9-18: Using letter-spacing

```
<block letter-spacing="0.2em">WELLINGTON NEW ZEALAND</block>
```

Figure 9-19: Text formatted using letter-spacing

It is possible to make letters closer than normal using a negative value for letter-spacing. Example FO for this is shown in Figure 9-20 and the result in Figure 9-21.

Figure 9-20: Moving letters closer together

```
<block letter-spacing="-0.1em">WELLINGTON NEW ZEALAND</block>
```

Figure 9-21: Text formatted using negative letter-spacing

9.11 Spacing before and after words

Spacing before and after text is specified using the space-start and space-end attributes on the inline element.

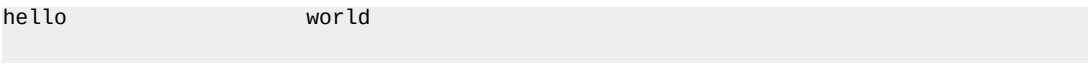
The space-start attribute specifies space to appear before text, space-end specifies space to appear after the text.

Figure 9-22 shows how to specify a gap between two words. This FO produces a 3cm gap between the words as shown in Figure 9-23 .

Figure 9-22: Using space-start

```
<block>
  hello <inline space-start="3cm">world</inline>
</block>
```

Figure 9-23: Output using space-start



Space between words is collapsed (i.e. merged) by default. If a word has space-end="1.0cm" and the following word has space-start="0.5cm", the gap between

the two words will be the larger of the two spaces (i.e. 1.0cm), not the sum. FO showing this is in Figure 9-24 and the output is in Figure 9-25.

Figure 9-24: FO showing merging of spaces

```
<block>
  <inline space-end="1cm">hello</inline>
  <inline space-start="0.5cm">world</inline>
</block>
```

Figure 9-25: The resulting 1.0cm space

```
hello      world
```

9.12 Forcing a line break

You can cause a line break in normal text by inserting an empty block element. Figure 9-26 shows an FO example which does this and Figure 9-27 shows the resulting output.

Figure 9-26: Forcing a line break

```
<block>
  this will be line one <block/>this will be line two
</block>
```

Figure 9-27: Line break created with an empty block

```
this will be line one
this will be line two
```

9.13 Space at the start of a line

Space specified with the space-start attribute is normally discarded at the start of the line. To force it to be retained use the space-start.conditionality attribute.

Figure 9-28 shows two blocks which create two lines. The first block will have no space at the start of the word. The second block has space-start.conditionality="retain" so the space specified by the space-start="1cm" will be retained. The output created by this FO is shown in Figure 9-29.

Figure 9-28: Using retain

```
<block background-color="#eeeeee">
  <inline space-start="1cm">
  discard
  </inline>
</block>
<block background-color="#eeeeee">
  <inline space-start="1cm" space-start.conditionality="retain">
  retain
  </inline>
</block>
```

Figure 9-29: Output from using retain

```
discard
      retain
```

9.14 Vertical alignment

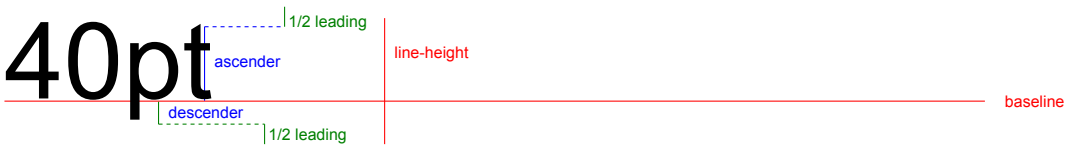
The vertical alignment of blocks of text within a containing flow or block is controlled by the display-align attribute.

The vertical alignment of words on a line is controlled by the vertical-align attribute.

Text on a line is positioned relative to the baseline, which is shown in Figure 9-30.

By default text sits on the baseline. In the terms of the XSL-FO specification, this is the *alphabetic* baseline.

Figure 9-30: The baseline



The height of the font above the baseline is the *ascender*. The height of the font below the baseline is the *descender*. Adding the ascender and descender values for the font (not for individual characters) gives the font size. The *leading* is the space above and below the characters, and is the difference between the line-height and the font-size.

The XSL-FO specification refers to the ascender value as the *text-altitude* and the descender as the *text-depth*. Together these two values add up to the *allocation rectangle* height. In these terms:

leading = (line-height - text-altitude - text-depth)

so

1/2 leading = (line-height - text-altitude - text-depth) / 2

By default the line height is 1.2em. The em unit is proportional to the size of the current font, so as the font size increases so does the line height. This can be changed by setting the Settings.LineHeightNormal value. For instance to make the line height larger and so space text out more vertically you could use the code in Figure 9-31.

Figure 9-31: Changing the default line height

```
FODocument doc = new FODocument();
doc.Settings.LineHeightNormal = "1.4em";
```

9.14.1 The effect of subscript and superscript text on line spacing

When calculating the largest characters on this line, we really mean those whose ascender and descender values are greatest (i.e. futherest from the baseline). When making this calculation, the value of the line-height-shift-adjustment attribute is considered. If text is a subscript or superscript and so has a baseline-shift value which changes its position vertically, this will also change its effective ascender and descender values. If line-height-shift-adjustment = "consider-shifts" (the default value) then the baseline-shift amount is taken into account when working out the greatest ascender and descender. If line-height-shift-adjustment = "disregard-shifts" then the effect of the baseline-shift is ignored. Setting line-height-shift-adjustment = "disregard-shifts" makes lines stay the same distance apart regardless of subscript and superscript elements.

The effect line-height-shift-adjustment is shown in Figure 9-32; the first two lines are in a block which has line-height-shift-adjustment= "consider-shifts" and so are further apart than the second two which are in a block which has line-height-shift-adjustment = "disregard-shifts":

Figure 9-32: Effect of disregard-shifts

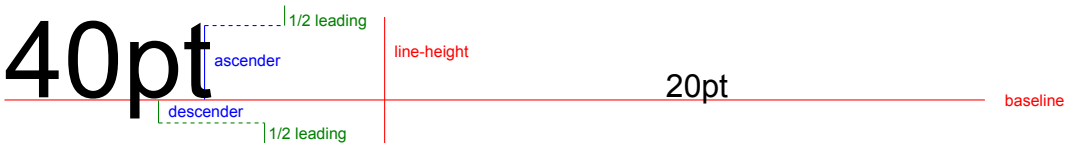
Specifies a string on which content of cells in a table column will align (see the section, in the CSS2 Recommendation[2]).

Specifies a string on which content of cells in a table column will align (see the section, in the CSS2 Recommendation[2]).

9.14.2 The baseline

The baseline is below the top of the text block a distance equal to 1/2 leading + max(ascender), which places the baseline in the same place for all text elements. This means that normally text rests on the same baseline regardless of the font size, as shown in Figure 9-33.

Figure 9-33: Text on the baseline

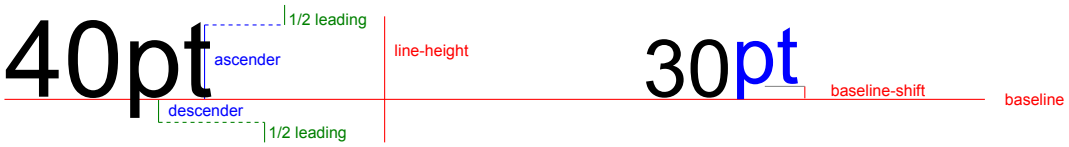


9.14.3 Subscript and superscript

Subscripted and superscripted text is created by using the baseline-shift attribute on an inline element.

The effect of the baseline shift is shown in Figure 9-34, where the "pt" characters are in an inline element with baseline-shift = "5pt".

Figure 9-34: Effect of baseline shift



The FO to move a word above the current baseline by 5 points is shown in Figure 9-35 with the resulting output appears in Figure 9-36.

Figure 9-35: FO showing baseline-shift

```
<block
  hello
  <inline color="red" baseline-shift="5pt">
    super
  </inline>
</block>
```

Figure 9-36: Output from the above FO

```
hello super
```

Font files contain default baseline shift values for superscripted and subscripted text. Rather than specifying baseline-shift="5pt", you can use the values "super" and "sub". The FO to move a word above the current baseline by the default amount for the current font is shown in Figure 9-37 with the resulting output in Figure 9-38. Using the "sub" and "super" values is preferable to using specific measurements because it means (a) if you change the font size of the paragraph you do not have to change all the baseline-shift values and (b) you get the baseline sift the font designer intended.

Figure 9-37: Using the default superscript

```
<block
  hello
  <inline color="red" baseline-shift="super">
    super
  </inline>
</block>
```

Figure 9-38: Output from the above FO

```
hello super
```

9.15 Line stacking strategies

XSL-FO uses the line-stacking-strategy attribute to determine how lines are stacked vertically on a page. The default value of this attribute is "max-height". When the "max-height" strategy is used the height of a line depends on the height of the characters or images on that line. The information which follows assumes that this default value is used. The other values for line-stacking-strategy, namely "font-height" and "line-height" will produce different results, since the height of the line using these strategies does not change when the content of the line changes.

The leading value is calculated from the line-height and font-size specified for the block element which contains the text. It is constant for the whole block and is not affected by other values specified on contained within the block.

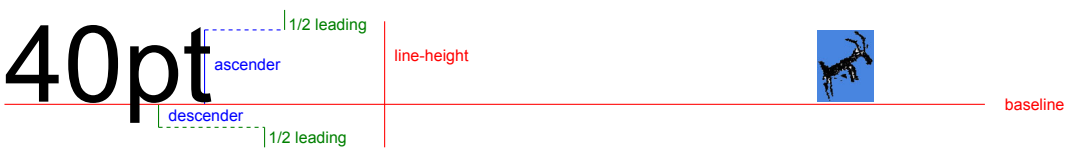
The height the line is calculated using "largest" characters found on the line, i.e. the sum of the max(ascender) and max(descender) values.

9.16 Aligning images

An inline element such as external-graphic is treated similarly to a text element. The height of the image is used as the ascender value. The descender value is zero.

This means that by default an image will be positioned on the baseline, as shown in Figure 9-39 .

Figure 9-39: An image on the baseline



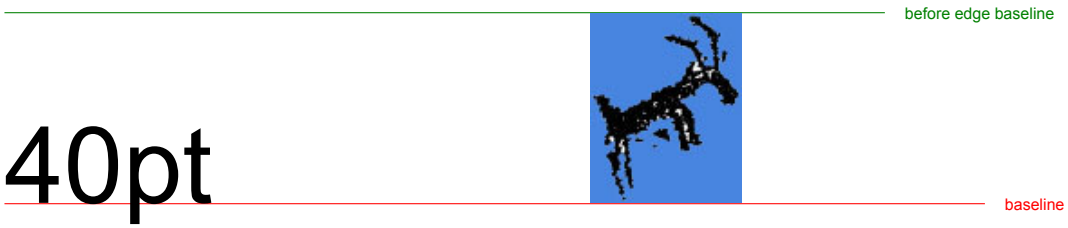
A large image will contribute a large ascender value to the baseline placement calculation, but will still sit on that baseline as shown in Figure 9-40.

Figure 9-40: Large image on baseline

9.16.1 The before-edge baseline

By default an element has an alignment-baseline value of "baseline" and so sits on the baseline shown in the above diagrams. For a given line, the largest thing on that line which has alignment-baseline = "baseline" establishes the position of the *before edge baseline*. This is shown in Figure 9-41.

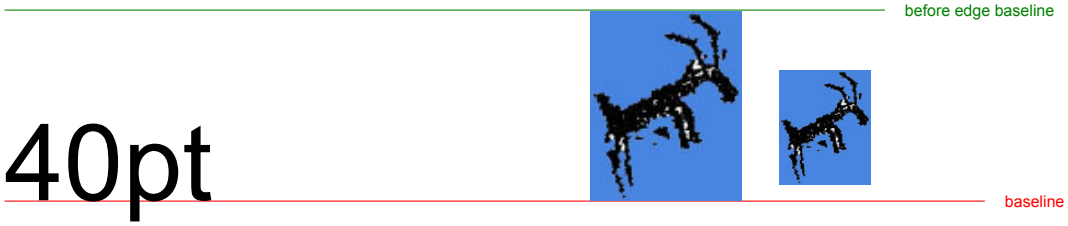
Figure 9-41: Image aligned to before-edge baseline



To align another object with the before edge baseline, either set vertical-align = "top" or alignment-baseline = "before-edge".

Figure 9-42 shows a second smaller image with default alignment, which positions it on the baseline.

Figure 9-42: Differently aligned images



By specifying vertical-align="top" on the external-graphic for the second image, we can align this image to the before edge baseline and get the layout shown in Figure 9-43.

Figure 9-43: Two images aligned using vertical-align

If all the elements on the line have vertical-align = "top", then the *before edge baseline* cannot be calculated, so the *text before edge baseline* is used. This is the top of the ascender for the font specified for the block which contains the elements.

Chapter 10

Fonts

Ibex supports TrueType and Type 1 (Postscript) fonts. Font information is read from the registry at runtime, no configuration of fonts is required.

Information on how to list the fonts which Ibex can use can be found in the usage chapter on page 30.

Ibex reads the registry to see which fonts are available. Specifically the entries under "HKLM\software\microsoft\windows nt\currentversion\fonts" list available fonts, and those under "HKLM\software\microsoft\windows nt\currentversion\fontsubstitutes" list translations from font names to existing fonts. Any of the font names listed in these two places can be used.

In addition Type 1 font names are read from "HKLM\software\microsoft\windows nt\currentversion\type 1 installer\type 1 fonts". Only Type 1 fonts that come as a PFM (metrics) and PFB (binary) pair of files are supported.

10.1 How Ibex uses fonts

Your FO file contains a series of letters. Each of which is stored in the file as a one or two byte *code point* such as 65 for 'A' or 0x8226 for the bullet character.

Ibex reads the TrueType or Type 1 font file and looks in the font to see if the font supports that particular code point. If it does, then the font maps that code point to a glyph, which is what gets displayed.

Not all fonts support all code points. For example arial.ttf is 370 KB in size, whereas arialuni.ttf is 23,000 KB, because arialuni has glyphs for a many more code points that arial.ttf.

Not all fonts map a code point to the same glyph. Some fonts map code points they do not support to a glyph such as the square box one.

Chapter 11

Floats

The float element can be used to position an image or other elements to the side or top of the page and cause text to flow around that image.

The paragraph in Figure 11-1 uses two float elements to make the image appear on the left and right sides, with the text flowing around the images below them.

Figure 11-1:
Left and right floats

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis in dolor quis lectus cursus condimentum. Vestibulum id nunc vitae dui semper ultrices. Mauris a mi. Phasellus eu lacus. Pellentesque eu ligula mattis odio faucibus faucibus. Aliquam sit amet nunc laoreet tellus ullamcorper malesuada. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Cras nec mauris. Proin cursus tincidunt leo. Maecenas metus lacus, imperdiet fermentum, blandit at, sollicitudin eu, sem. Duis elementum libero vitae lacus. Curabitur justo. Aliquam erat volutpat. Maecenas nec nulla in massa consectetuer volutpat. Aenean turpis nisl, rutrum a, posuere sit amet, varius in, enim. Praesent risus.

Nam volutpat enim eget neque. Maecenas a dui ac felis nonummy sollicitudin. Proin iaculis. Vestibulum in eros sit amet libero mollis convallis. nunc laoreet tellus ullamcorper malesuada. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Cras nec mauris. Proin cursus tincidunt leo. Maecenas metus lacus, imperdiet fermentum, blandit at, sollicitudin eu, sem. Duis elementum libero vitae lacus. Curabitur justo. Aliquam erat volutpat. Maecenas nec nulla in massa consectetuer volutpat. Aenean turpis nisl, rutrum a, posuere sit amet, varius in, enim. Praesent risus. Nam volutpat enim eget neque. Maecenas a dui ac felis nonummy sollicitudin. Proin iaculis. Vestibulum in eros sit amet libero mollis convallis.

This effect is achieved by having a block which contains the text and two float elements. The float elements in turn contain a block-container element which has a inline-progression-dimension attribute defining the width of the float area. Any elements inside the block-container will be in the float area. If a block-container is not used within the float and the width of the float cannot be determined, a default configurable value is used.

The FO for creating the above is show in Figure 11-2. Figure 11-2 is itself contained inside a float with float = "before", which will make it appear at the top of the following page. This technique is used in this manual when we do not want a large example to be split across page breaks or to interrupt the content. When a float has float = "before", its position in the PDF file is not the same as its position in the FO file, in that it will be moved to the top of the next page and the blocks before and after the float will flow as if the float was not there.

The side on which the float occurs is specified using the float attribute. This can be set to "left" or "right" to position the float at the side of the page. It can also be set to "before" to position the float at the start of the next page.

Figure 11-2:
FO for float example

```
<block font-size="1.0em" text-align="justify">
    <float float="left">
      <block-container inline-progression-dimension="2.5cm">
        <block text-align="center">
          <external-graphic src="url(ibexorange.jpg)" content-width="50%"
               padding="3pt"/>
        </block>
      </block-container>
    </float>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis in dolor quis lectus
cursus condimentum. Vestibulum id nunc vitae
dui semper ultrices. Mauris a mi. Phasellus eu lacus. Pellentesque eu ligula mattis
odio faucibus faucibus. Aliquam sit amet
    <float float="right">
      <block-container inline-progression-dimension="2.5cm">
        <block text-align="center">
          <external-graphic src="url(ibexorange.jpg)" content-width="50%"
              padding="3pt"/>
        </block>
      </block-container>
    </float>
nunc laoreet tellus ullamcorper malesuada. Vestibulum ante ipsum primis in faucibus
orci luctus et ultrices posuere cubilia Curae; Cras nec mauris. Proin cursus
tincidunt leo. Maecenas metus lacus, imperdiet fermentum, blandit at,
sollicitudin eu, sem. Duis elementum libero vitae lacus. Curabitur justo. Aliquam
erat volutpat. Maecenas nec nulla in massa consectetuer volutpat. Aenean turpis
nisl, rutrum a, posuere sit amet, varius in, enim. Praesent risus. Nam volutpat
enim eget neque. Maecenas a dui ac felis nonummy sollicitudin. Proin iaculis.
Vestibulum in eros sit amet libero mollis convallis. nunc laoreet tellus
ullamcorper malesuada. Vestibulum ante ipsum primis in faucibus orci luctus et
ultrices posuere cubilia Curae; Cras nec mauris. Proin cursus tincidunt leo.
Maecenas metus lacus,imperdiet fermentum, blandit at, sollicitudin eu, sem.
Duis elementum libero vitae lacus. Curabitur justo. Aliquam erat volutpat.
Maecenas nec nulla in massa consectetuer volutpat. Aenean turpis nisl, rutrum a,
posuere sit amet, varius in, enim. Praesent risus. Nam volutpat enim eget neque.
Maecenas a dui ac felis nonummy sollicitudin. Proin iaculis. Vestibulum in eros
sit amet libero mollis convallis.
</block>
```

Side floats (with float = "left" or float = "right") are closely tied to the block which contains the float element. If the float element does not fit on the page, then the float and some or all of the containing block will be moved to the following page. This ensures that the text in the block does not refer to (for example) an image in the float which is not on the same page as the text.

11.1 How the float width is calculated

Ibex looks at the content of the float element to try and determine how wide the float should be. If a block-container element is found directly below the float element, and this block-container has a width attribute, then that determines the width of the float. If no width can be found, then the width of the float is calculated from by multplying the containing block width by Settings.SideFloatDefaultWidthPercentage, which defaults to 30%.

Chapter 12

Space Handling

XSL-FO defines various attributes for managing whitespace in FO. These allow you to control how linefeeds and whitespace are output.

12.1 Linefeeds and carriage returns

A linefeed is a character with ASCII code 10, or Unicode code point U+000A. This is different to a carriage return which has ASCII code 13. Ibex acts on linefeeds, not on carriage returns. Carriage returns are ignored during PDF creation.

12.2 Default treatment of linefeeds and spaces

By default linefeeds and whitespace preceding and following linefeeds are removed during formatting. Figure 12-1 shows FO which has linefeeds at the end of each line. The resulting output shown in Figure 12-2 has neither linefeeds nor spaces around the text. This is the default treatment for text in XSL-FO.

Figure 12-1: Text with linefeeds and spaces

```
<block margin='2cm'>To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
</block>
```

Figure 12-2: Output with default handling

```
To be, or not to be: that is the question: Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune, Or to take arms against a sea of
troubles,
```

12.3 Using linefeeds to break text

The linefeed-treatment attribute is used to specify the treatment of linefeeds in text. This defaults to "ignore" causing linefeeds to be ignored. We can retain the linefeeds by setting the linefeed-treatment attribute to "preserve". Figure 12-3 shows our example with this attribute added. Figure 12-4 shows the output from this FO.

Figure 12-3: Using linefeed-treatment

```
<block linefeed-treatment="preserve">To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
</block>
```

Figure 12-4: Output with linefeeds preserved

```
To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
```

12.4 Retaining spaces

The white-space-treatment and white-space-collapse attributes are used to control the handling of spaces.

If we want to put some formatted code in our documen, Figure 12-5 shows FO for this.

Figure 12-5: Code example

```
<block linefeed-treatment="preserve">
private void swap_byte( ref byte x, ref byte y ) {
    byte t = x;
    x = y;
    y = t;
}
</block>
```

Setting linefeed-treatment = "preserve" we get the output show in Figure 12-6. We have preserved the linefeeds but all formatting spaces have gone.

Figure 12-6: Code with linefeeds but no spacing

```
private void swap_byte( ref byte x, ref byte y ) {
byte t = x;
x = y;
y = t;
}
```

The white-space-collapse attribute controls whether Ibex compresses adjacent white space characters into a single space. By default any number of adjacent spaces are compressed into a single space.

The white-space-treatment attribute controls whether Ibex ignores spaces adjacent to linefeeds. Setting white-space-treatment = "preserve" makes Ibex retain white space which appears adjacent to linefeeds.

If we set white-space-treatment to "preserve", and white-space-collapse to "false" we will retain the white spaces around the linefeeds. The FO for this is shown in Figure 12-7, and the formatted output is shown in Figure 12-8.

Figure 12-7: FO to retain spaces and linefeeds

```
<block
   linefeed-treatment="preserve"

   white-space-treatment="preserve"
   white-space-collapse="false"
      >
private void swap_byte( ref byte x, ref byte y ) {
```

```
    byte t = x;
    x = y;
    y = t;
}
</block>
```

Figure 12-8: Output with linefeeds but no spacing

```
private void swap_byte( ref byte x, ref byte y ) {
    byte t = x;
    x = y;
    y = t;
}
```

12.5 Non-breaking spaces

Unicode defines the code point U+00A0 called NO-BREAK SPACE. This can be used to insert a space between words without allowing a line break to occur between the words. Ibex treats two words separated by a U+00A0 as a single word.

The non-breaking space can be inserted into XML using the   entity.

The example in Figure 12-9 shows a block used in a table header. It contains the three words "Score per 100". The default formatting is shown in Figure 12-10. If we want to move the word "per" to the next line to keep it with the "100", we replace the space between "per" and "100" with a non-breaking space. This will prevent Ibex breaking the line between the "per" and "100" words.

Figure 12-11 shows the FO with a non-breaking space and Figure 12-12 shows the resulting output.

Figure 12-9: FO without a non-breaking space

```
<block-container width="2.8cm">
  <block border="1pt solid black"
     padding="3pt" text-align="center">
    Score per 100
  </block>
</block-container>
```

Figure 12-10: Output without a non-breaking space

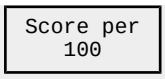
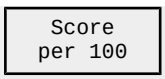


Figure 12-11: FO with non-breaking space

```
<fo:block-container width="2.8cm">
  <fo:block border="1pt solid black"
       padding="3pt" text-align="center">
    Score per 100
  </fo:block>
</fo:block-container>
```

Figure 12-12: Output with a non-breaking space



Chapter 13

Colors

XSL-FO defines various attributes for managing color. By default a block is displayed with the foreground color (that is the text) being black and the background color being white.

Colors are most commonly expressed using the RGB color scheme, where there are three parts to a color: red, green and blue. Ibex also supports the CMYK color scheme commonly used in the printing industry.

13.1 Text color

The color of text is specified using the color attribute. Figure 13-1 shows a simple example of some FO to make text blue. The output is shown in Figure 13-2.

Figure 13-1: FO for blue text

```
<block color="blue">
  To be, or not to be: that is the question:
</block>
```

Figure 13-2: Blue text

```
The resulting text will be blue like this
```

13.2 Background color

The background color of any element is defined using the background-color attribute. Figure 13-3 shows FO for a block with a gray background. The output from this is shown in Figure 13-4.

Figure 13-3: FO for gray background

```
<block background-color="gray">
    To be, or not to be: that is the question:
</block>
```

Figure 13-4: Gray background

```
The resulting text will have a gray background like this
```

13.3 Available colors

The value used for the color and background-color attributes can be a predefined color such as "red", an RGB color defined using a hex value such as "#eeffdd" or a CMYK color.

13.3.1 Predefined colors

XSL-FO uses the list of colors defined for HTML 4.0, which contains these values:

aqua	ibex
black	ibex
blue	ibex
fuchsia	ibex
gray	ibex
green	ibex
lime	ibex
maroon	ibex
navy	ibex
olive	ibex
purple	ibex
red	ibex
silver	ibex
teal	ibex
white	
yellow	ibex

13.3.2 Hexadecimal RGB colors

A color can be defined as a string of six digits preceded by a "#" character. The first two digits define the red component of the color, in a range from 0 to 255. The second two digits define the green component and the last two digits define the blue component. This is the same scheme for defining colors as is used in HTML.

13.3.3 CMYK colors

CMYK colors are four-part colors using values for cyan, magenta, yellow and black respectively. The CMYK system is *subtractive*, meaning that higher values mean less color, unlike RGB where higher values mean more color. CMYK colors are used in the printing industry to define a color which will appear the same across all media. Typically a color defined using RGB will not appear exactly the same on the screen and on a

printed page, or even on two different computer screens. CMYK colors are used to ensure that colors are the same on screen and on the printed page.

PDF files are usually created with single color scheme. You would not usually mix CMYK and RGB colors in one document. Note that when creating a CMYK PDF file any images included in the document should be in CMYK format.

A CMYK color is defined using the rgb-icc() function. This takes eight parameters. The first three define the red, green and blue components of a fallback RGB color, the fourth defines the color profile name, and the last four define the four parts of the CMYK color. The color profile must have been declared in the declarations formatting object using a color-profile element.

Figure 13-5 shows an example of the rgb-icc() function.

Figure 13-5: The rgb-icc function

```
<block color="rgb-icc( 0, 0, 0, cmyk, 0.7,0.3,0.3,0.4 )">
   in cmyk .5,.5,.5,0
</block>
```

In Figure 13-5 the three components of the fallback RGB color are zero. This is normal because we are creating a CMYK PDF file and will not be using any fallback RGB colors. The color profile name is "cmyk". Ibex requires that the color profile name be "cmyk" when creating a CMYK color.

A complete document using the CMYK color space is shown in Figure 13-6. This shows how to use the declarations and color-profile elements to define a color profile.

Figure 13-6: FO for a CMYK PDF file

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
 <layout-master-set>
  <simple-page-master master-name="page">
   <region-body margin="1in"
       region-name="body"/>
  </simple-page-master>
 </layout-master-set>

 <declarations>
     <color-profile src="src"
            color-profile-name="cmyk"/>
 </declarations>

 <page-sequence master-reference="page">
  <flow flow-name="body">
   <block color="rgb-icc( 0, 0, 0, cmyk, 0.7,0.3,0.3,0.4 )">
   in cmyk .5,.5,.5,0
   </block>
  </flow>
 </page-sequence>
</root>
```

13.3.4 PDF/X color profiles

Ibex can create PDF files which conform to the PDF/X standard. These files can include embedded color profiles, used to define a common color scheme across different devices.

Color profiles are loaded from files on disk and included in the PDF file. Some color profiles are very large (i.e. > 500k) and can result in large PDF files.

Loading a color profile from a file on disk is an Ibex extension. The name of the color profile file is specified using the color-profile-file-name attribute of the ibex:pdfx element, as shown in Figure 13-7 below.

Figure 13-7: FO for a PDF/X showing the loading of a color profile

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.w3.org/1999/XSL/Format"
xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">
 <layout-master-set>
  <simple-page-master master-name="page" page-width="20cm">
   <region-body region-name="body" margin="3cm" reference-orientation='0'/>
  </simple-page-master>
 </layout-master-set>

 <ibex:pdfx color-profile-file-name="colorprofiles\USWebCoatedSWOP.icc"
output-condition="TR001 SWOP/CGATS"/>

 <page-sequence master-reference="page">
  <flow flow-name="body">

   <block font="10pt arial">
    hello world
   </block>
 </flow>


 </page-sequence>
</root>
```

Chapter 14

Lists

Lists are created using the list-block element. A list-block in XSL-FO is an area of content divided into two columns.

A simple list-block is shown in Figure 14-1. The list created by this FO is shown in Figure 14-2.

Figure 14-1: FO for a list

```
<list-block provisional-distance-between-starts=".5cm"
   provisional-label-separation="0.1cm">
  <list-item>
     <list-item-label end-indent="label-end()">
        <block font='8pt arial'>&#x25CF;</block>
     </list-item-label>
     <list-item-body start-indent="body-start()">
          <block>
              item one
          </block>
      </list-item-body>
  </list-item>
  <list-item>
     <list-item-label end-indent="label-end()">
        <block font='8pt arial'>&#x25CF;</block>
     </list-item-label>
     <list-item-body start-indent="body-start()">
          <block>
              item two
          </block>
      </list-item-body>
 </list-item>
</list-block>
```

Figure 14-2: A list

- item one
- item two

Features of lists include:

- the list-block is a block-level element which contains the whole list.
- the provisional-distance-between-starts attribute on the list-block defines the distance between the start of the label and the start of the body.
- the provisional-label-separation attribute on the list-block defines the size of the gap between the end of the label and the start of the body. This gap is created by reducing the size of the label. For example, if provisional-distance-between-starts is 5cm and the provisional-label-separation is 1cm, then the start edges of the label and body will be 5cm apart, and the label will be 4cm (5cm - 1cm) wide.

- each item in the list is contained in a list-item element.
- each list-item must contain both a list-item-label and a list-item-body. The list-item-label must come first.
- the list-item-label should have the end-indent attribute set to "label-end()". This is a special function which returns a value derived from provisional-distance-between-starts and provisional-label-separation.
- the list-item-body should have the start-indent attribute set to "body-start()". This is a special function which returns a value derived from provisional-distance-between-starts and provisional-label-separation.
- both the list-item-label and list-item-body contain one or more block-level elements, so a list-item-label or list-item-body can contain other block-level elements such as block, table and list-block.

14.1 Bulleted lists

The example in Figure 14-1 also shows how to insert a Unicode character into the FO, using the syntax ●.

This table shows some common bullet types for lists:

Unicode	Result
•	•
‣	‣
●	●
○	○
■	■
□	□
◆	◆
◇	◇

Note that what is displayed in the document depends on whether the font you are using contains the specified character. If the font does not contain the specified character you will see a warning message like the one in Figure 14-3.

```
warning:380 No glyph index found for code point 2023 in font ArialMT
```

Figure 14-3: Error message if bullet not in font

Chapter 15

Tables

A table in XSL-FO is an area of content divided into rows and columns. A table is created with the table element.

A FO for a simple table is shown in Figure 15-1 and the output it creates is shown in Figure 15-2. This shows the basic structure of a table element containing table-body, table-row and table-cell elements.

Figure 15-1: FO for a simple 2 x 2 table

```
<table>
  <table-body>
    <table-row>
      <table-cell border="1pt solid blue" padding="2pt">
            <block>row 1 column 1</block>
      </table-cell>
      <table-cell border="1pt solid blue" padding="2pt">
            <block>row 1 column 2</block>
      </table-cell>
    </table-row>
    <table-row>
       <table-cell border="1pt solid blue" padding="2pt">
             <block>row 2 column 1</block>
       </table-cell>
       <table-cell border="1pt solid blue" padding="2pt">
             <block>row 2 column 2</block>
       </table-cell>
    </table-row>
</table-body>
```

Figure 15-2: The simple 2 x 2 table

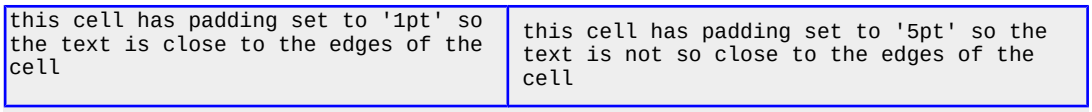
row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

The padding and border attributes are not inherited from containing elements, so are best defined on the table-cell elements.

15.1 Cell padding

Padding is the amount of space that appears between the inside edge of the border of a cell and the outside edge of the content of the cell. Padding is specified by the padding attribute. The default amount of padding is '0pt'. Figure 15-3 shows a table with two cells. The first cell has padding="1pt" and the second has padding="5pt". Padding is almost always used to avoid having the content too close to the cell borders.

Figure 15-3:
FO showing cells with different padding


this cell has padding set to '1pt' so the text is close to the edges of the cell
this cell has padding set to '5pt' so the text is not so close to the edges of the cell

The padding attribute sets padding for all four sides of the cell. Individual sides can be set using the padding-left, padding-right, padding-top and padding-bottom attributes.

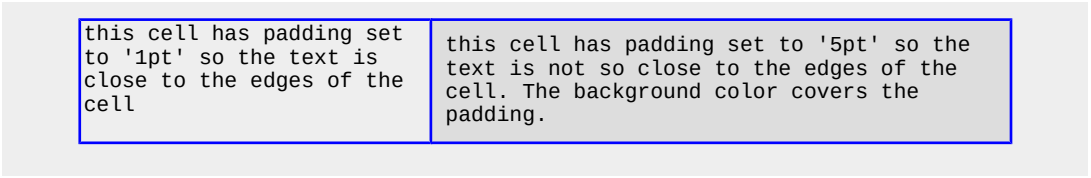
The padding attribute also supports a shorthand format where:

- if one value is specified (padding="2pt") the same value will apply to all four sides;
- if two values are specified (padding="2pt 3pt") the first value will apply to the top and bottom edges, the second value to the left and right edges;
- if three values are specified (padding="2pt 3pt 1pt") the first value will apply to the top edge, the second to the left and right edges, and the third to bottom edge;
- if four values are specified (padding="2pt 3pt 1pt 0pt") these will apply to top, right, bottom and left edges in that order.

15.2 Cell background color

The background color of a cell is specified using the background-color attribute. This supports the same predefined colors as CSS and the use of hex values such as "#33ffcc". The background color of the cell extends to the inside edge of the border, which means that the area specified by the padding attribute is colored by the background color. This is shown in Figure 15-4 where the second cell has the attribute background-color = "#dddddd".

Figure 15-4:
Cell with background color set


this cell has padding set to '1pt' so the text is close to the edges of the cell
this cell has padding set to '5pt' so the text is not so close to the edges of the cell. The background color covers the padding.

If you do not want the background to extend to the edge of the padding, specify the background-color attribute on the contents of the cell (i.e. the block elements) rather than on the table-cell. An example FO for this is shown in Figure 15-5 and the resulting output appears in Figure 15-6.

Figure 15-5: FO setting the background color on a block

```
<table>
  <table-body>
    <table-row>
      <table-cell border='1pt solid blue' padding='1pt'>
            <block>
            this cell has padding set to '1pt' so the text is close to the edges of
the cell
            </block>
      </table-cell>
      <table-cell border='1pt solid blue' padding='5pt'
            background-color='#dddddd'>
            <block background-color='#dddddd'>
            this cell has padding set to '5pt' so the text is not so close to the
edges of the cell
            </block>
      </table-cell>
    </table-row>
</table-body>
```

Figure 15-6: Cell with background color on the block element

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has padding set to '5pt' so the text is not so close to the edges of the cell

15.3 Cell background images

An image can be used as the background to a cell by specifying the background-image element, as shown in Figure 15-7. This produces the output shown in Figure 15-8.

Figure 15-7: FO for using an image as a cell background

```
<table>
  <table-body>
    <table-row>
      <table-cell border='1pt solid blue' padding='1pt'>
            <block>
            this cell has padding set to '1pt' so the text is close to the edges of
the cell
            </block>
      </table-cell>
      <table-cell border='1pt solid blue' padding='5pt'
            background-image='url(ibex.jpg)'>
            <block>
            this cell has a background image
            </block>
      </table-cell>
    </table-row>
</table-body>
```

Figure 15-8: Cell with image background

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has a background image

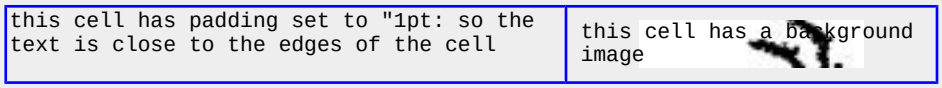
As the above example shows, by default the image will be repeated if it is less than the width of the cell. This can be changed using the background-repeat attribute. If this is set to "no-repeat" the output changes to that shown in Figure 15-9.

Figure 15-9: Using background-repeat = 'no-repeat'

this cell has padding set to '1pt' so the text is close to the edges of the cell	this cell has a background image

The background image can be positioned in the cell using the background-position-horizontal and background-position-vertical attributes. The cell in Figure 15-10 example has background-position-horizontal set to "50%".

Figure 15-10: Centering the background image



15.4 Implicit and explicit rows

Usually FO files use the table-row element to define which cells are in which rows, as shown in Figure 15-11.

Figure 15-11: Tables with cells contained in rows

```
<table>
  <table-body>
    <table-row>
      <table-cell border="1pt solid blue" padding="2pt">
            <block>row 1 column 1</block>
      </table-cell>
      <table-cell border="1pt solid blue" padding="2pt">
            <block>row 1 column 2</block>
      </table-cell>
    </table-row>
    <table-row>
       <table-cell border="1pt solid blue" padding="2pt">
             <block>row 2 column 1</block>
       </table-cell>
       <table-cell border="1pt solid blue" padding="2pt">
             <block>row 2 column 2</block>
       </table-cell>
    </table-row>
</table-body>
```

It is possible to dispense with the table-row element and have the table-body contain table-cell elements directly. In this case any cell can have the ends-row attribute set to "true", which causes a new row to be started containing the next cell. This approach is sometimes easier to use when generating the FO using XSLT.

Figure 15-12 shows what the above FO would look like if we changed it to use implicit rows. The output from this appears in Figure 15-13 below.

Figure 15-12: FO for a table with implicit rows

```
<table>
  <table-body>
      <table-cell border='1pt solid blue' padding='2pt'>
            <block>row 1 column 1</block>
      </table-cell>
      <table-cell border='1pt solid blue' padding='2pt'
            ends-row='true'>
            <block>row 1 column 2</block>
      </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 2</block>
       </table-cell>
</table-body>
```

Figure 15-13: Table with implicit rows

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

15.5 Table columns

The table-column element is used to set the column width and other characteristics of a table column. A table-column element has an associated column number which determines which column the table-column element refers to. This column number is either implied (with the first table-column element applying to the first column, the second to the next etc.), or explicitly set using the column-number attribute.

A single table-column element can be used to define the style of multiple columns by using the number-columns-spanned attribute.

Figure 15-14 shows the FO for a table with two table-column elements, which apply to the first and second columns. In this case they set the column widths (to 30% and 70%), and the give the second column a shaded background. The output created from the FO appears in Figure 15-15.

Figure 15-14: FO using table-column elements

```
<table>
    <table-column column-width='30%'/>
      <table-column column-width='70%'
         background-color='#dddddd'/>
  <table-body>
    <table-row>
      <table-cell border='1pt solid blue' padding='2pt'>
            <block>row 1 column 1</block>
      </table-cell>
      <table-cell border='1pt solid blue' padding='2pt'>
            <block>row 1 column 2</block>
      </table-cell>
    </table-row>
    <table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 2</block>
       </table-cell>
    </table-row>
</table-body>
```

Figure 15-15: Table with defined column widths

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

Some cell attributes such as background color are determined using attributes from the cell itself and from the other elements of the table structure. The order of precedence in determining cell characteristics such as background-color is table-cell, table-row, table-body, table-column and finally table.

15.6 Proportional column widths

Columns can be allocated widths which are proportional to the widths of other columns. For example, if we have two columns and want to give the first column twice the width

of the second, we can specify column widths using the proportional-column-width() function as shown in Figure 15-16. The total of the values used in the proportional-column-width() functions is 3 (2+1), so the first column will gave 2/3 of the width and the second 1/3. The output from this FO appears in Figure 15-17.

Figure 15-16: FO using proportional column widths

```
<table>
  <table-column
        column-width='proportional-column-width(2)'/>
  <table-column
        column-width='proportional-column-width(1)'
        background-color='#dddddd'/>
  <table-body>
    <table-row>
      <table-cell border='1pt solid blue' padding='2pt'>
            <block>row 1 column 1</block>
      </table-cell>
      <table-cell border='1pt solid blue' padding='2pt'>
            <block>row 1 column 2</block>
      </table-cell>
    </table-row>
    <table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 1</block>
       </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 2</block>
       </table-cell>
    </table-row>
</table-body>
```

Figure 15-17: Output from proportional width example

row 1 column 1	row 1 column 2
row 2 column 1	row 2 column 2

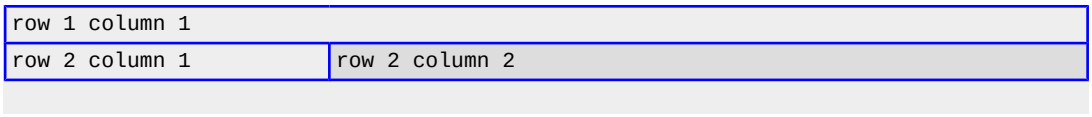
15.7 Spanning columns and rows

The number of columns which a cell spans is set by the number-columns-spanned attribute. An example FO for this is shown in Figure 15-18. In this example the first cell of the first row spans two columns. The output from this FO appears in Figure 15-19.

Figure 15-18: FO for cell spanning 2 columns

```
<table>
  <table-column column-width="30%"/>
  <table-column column-width="70%"
        background-color="#dddddd"/>
  <table-body>
    <table-row>
      <table-cell border="1pt solid blue" padding="2pt"
          number-columns-spanned="2">
            <block>row 1 column 1</block>
      </table-cell>
    </table-row>
    <table-row>
       <table-cell border="1pt solid blue" padding="2pt">
             <block>row 2 column 1</block>
       </table-cell>
       <table-cell border="1pt solid blue" padding="2pt">
             <block>row 2 column 2</block>
       </table-cell>
    </table-row>
</table-body>
```

Figure 15-19: Cell spanning two columns

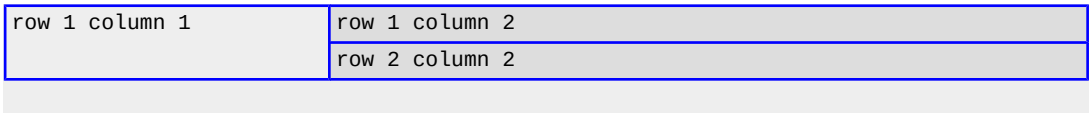


The number of rows which a cell spans is set by the number-rows-spanned attribute. Example FO for this is shown in Figure 15-20. In this example the first cell of the first row spans two rows. The output from this FO appears in Figure 15-21.

Figure 15-20: FO for cell spanning two rows

```
<table>
    <table-column column-width='30%'/>
    <table-column column-width='70%'
         background-color='#dddddd'/>
    <table-body>
    <table-row>
      <table-cell border='1pt solid blue' padding='2pt'
          number-rows-spanned='2'>
            <block>row 1 column 1</block>
      </table-cell>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 1 column 2</block>
       </table-cell>
    </table-row>
    <table-row>
       <table-cell border='1pt solid blue' padding='2pt'>
             <block>row 2 column 2</block>
       </table-cell>
    </table-row>
</table-body>
```

Figure 15-21: Output for cell spanning two rows



15.8 Cell separation

XSL-FO has two ways of processing the borders of adjacent cells depending on the value of the border-collapse attribute on the table.

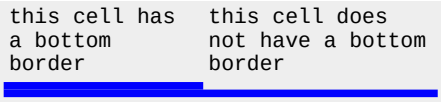
If border-collapse="collapse", which is the default, there is no gap between cells and the borders of adjacent cells are merged (or "collapsed") to get a single border shared by both cells. The rules for combining borders are explained in the XSL-FO specification. Broadly speaking the widest border will be used. This is called the collapsed border model.

If border-collapse="separate" adjacent borders are not merged. A gap can be inserted between adjacent borders using the border-spacing attribute. The border-spacing attribute can have one or two values. If one value is specified (for instance border-spacing="1mm") the vertical and horizontal spacing between cells is set to this value. If two values are specified separated by a space (for instance border-spacing='1mm 3mm') the horizontal separation is set to the first value and the vertical separation is set to the second. This is called the separated border model.

The following examples use a table with one row containing two cells. The first cell has a bottom border, the second does not. The table also has a bottom border.

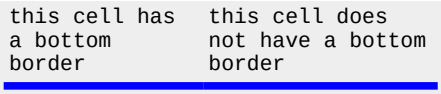
In the separate border model the border from the first cell will be drawn before the border of the table as shown in Figure 15-22.

Figure 15-22: Cells with separate borders



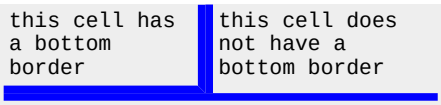
In the collapsed border model the border from the first cell will be merged with the border of the table and a single border will be drawn as shown in Figure 15-23.

Figure 15-23: Cell border collapsed with table border



If we add an inner border to each cell we can see this with the separate model, as shown in Figure 15-24.

Figure 15-24: Separate cell and table borders



With the collapsed border model the border between the two cells will be half the width it is in the separate model, as shown in Figure 15-25.

Figure 15-25: Collapsed borders

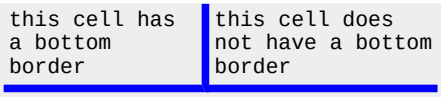


Figure 15-26 shows an example of a table with separate borders. Note how the border-spacing on the previous table sets the space between cells only, not the space between the cell and the table border. This space can be set using padding. If we add padding="2mm" to the table we get the layout shown in Figure 15-27.

Figure 15-26: Table with separate borders

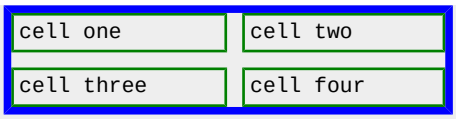
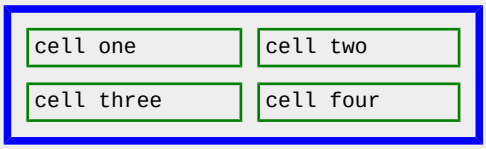


Figure 15-27: Cells separated from the table borders by padding



15.9 Table headers

Table headers are created using the table-header element. The table-header should appear inside the table element after any table-column elements and before any table-body elements. The table-header element is similar in structure to a table-body element in that it contains table-row elements.

This section describes the behavior of table headers which do not change. Headers which can have different content on different pages are described later in this chapter in the section on continuation markers on page 91.

Figure 15-28 shows the FO for a simple table with a one row header and two content rows. The output created by the FO appears in Figure 15-29.

Figure 15-28: Simple table with header

```
<table>
 <table-column column-width="100%"/>
 <table-header>
  <table-row>
    <table-cell border="1pt solid black" padding="5pt">
     <block>Heading</block>
    </table-cell>
  </table-row>
 </table-header>
 <table-body>
  <table-row>
    <table-cell border="1pt solid black" padding="5pt">
     <block>row 1</block>
    </table-cell>
  </table-row>
  <table-row border="1pt solid black" padding="5pt">
    <table-cell>
     <block>row 2</block>
    </table-cell>
  </table-row>
 </table-body>
</table>
```

Figure 15-29: Table with simple header

Heading
row 1
row 2

Table headers are repeated at the top of the table after each page break. This is the default. To prevent the table header appearing on pages after the first, specify table-omit-header-at-break = "true" on the table element.

15.10 Table footers

Table footers are created using the table-footer element. The table-footer should appear inside the table element after any table-column and table-header elements and before any table-body elements. The table-footer element is similar in structure to a table-body element in that it contains table-row elements.

It is a common error to place the table-footer element at the end of the table, after the table-body elements. It must be placed before the table-body elements because Ibex may start rendering the table to PDF before the whole table has been read from the FO file.

This section describes the behavior of table footers which do not change. Footers which can have different content on different pages are described later in this chapter in the section on continuation markers on page 91.

Figure 15-30 shows the FO for a simple table with a one row header and footer and two content rows. The output created by the FO appears in Figure 15-31.

Figure 15-30: FO for simple table with header and footer

```
<table>
 <table-column column-width="100%"/>
 <table-header>
  <table-row>
    <table-cell border="1pt solid black" padding="5pt">
     <block>Heading</block>
    </table-cell>
  </table-row>
 </table-header>
 <table-footer>
  <table-row>
    <table-cell border="1pt solid black" padding="5pt">
     <block>Footer</block>
    </table-cell>
  </table-row>
 </table-footer>
 <table-body>
  <table-row>
    <table-cell border="1pt solid black" padding="5pt">
     <block>row 1</block>
    </table-cell>
  </table-row>
  <table-row border="1pt solid black" padding="5pt">
    <table-cell>
     <block>row 2</block>
    </table-cell>
  </table-row>
 </table-body>
</table>
```

Figure 15-31: Table with simple header and footer

Heading
row 1
row 2
Footer

Table footers are repeated at the bottom of the table before each page break. This is the default. To prevent the table footer appearing on pages other than the last, specify table-omit-footer-at-break = "true" on the table element.

15.11 Behavior at page breaks

15.11.1 Repeating headers

Table headers are repeated at the top of the table after each page break. This is the default. To prevent the table header appearing on pages after the first, specify table-omit-header-at-break = "true" on the table element.

15.11.2 Repeating footers

Table footers are repeated at the bottom of the table before each page break. This is the default. To prevent the table footer appearing on pages other than the last, specify table-omit-footer-at-break = "true" on the table element.

15.11.3 Repeating table borders

Table borders by default do not repeat at a break in the table, so the top border of a table is rendered only on the first page the table is on and the bottom border is rendered only on the last page.

To make the table bottom border repeat at each page break it is necessary to specify border-after-width.conditionality = "retain" on the table element.

To make the table top border repeat at each page break it is necessary to specify border-before-width.conditionality = "retain" on the table element.

15.12 Table continuation markers

Table continuation markers provide a way of dynamically changing the header and footer on a table so that different content can be displayed on different pages. A typical use of this feature is to put the words "continued on next page" in the footer of a table on all pages except the last.

Here we examine how the "continued on next page" requirement can be satisfied using Ibex. The approach taken by XSL-FO has two parts, implemented using the marker and retrieve-table-marker elements. First a retrieve-table-marker element is added to the footer. When the PDF is created this element will be replaced by the contents of one of the marker elements which has the same class name. The marker element which appears in the footer depends on the values of the attributes on the retrieve-table-marker.

The footer for this example is shown in Figure 15-32. As the PDF file is created the contents of the marker element with marker-class-name = "continued" will be located and inserted into the table-footer element. *The content of the marker element must be valid FO elements for their location in the table-footer*. In this example the retrieved elements go directly under the table-footer element, so the elements retrieved must be table-row elements.

Figure 15-32: FO for retrieve-table-marker

```
<table-footer>
  <retrieve-table-marker
        retrieve-class-name="continued"
        retrieve-position-within-table="first-starting"
        retrieve-boundary-within-table="page"/>
</table-footer>
```

Typically, there will be more than one marker element which has the marker-class-name = "continued". If this is not the case then the footer content will never change. The retrieve-position attribute specifies which marker to retrieve. In this example we want the first marker which appears on the page, so we use retrieve-position =

"first-starting-within-page". We also specify retrieve-boundary = "table" so any marker from any part of the table which has been output to PDF can be retrieved. Other options are detailed later in this section.

Conceptually, Ibex looks at every row in the table which has been output to the PDF file (including rows on the current page), collects all the markers associated with each of those rows and selects one to go into the footer. Markers associated with rows which are not on either the current page or prior pages are not considered. It is possible to have a different marker associated with every row in the table. This is useful for situations such as like rendering a running total.

The second part of the process is to define one or more marker elements. In this case our marker elements are associated with table-row elements. The first table-row has a marker element which specifies the "continued on next page" text. The contents of this marker will be retrieved for all pages except the last.

The last row of the table has an empty marker element. The content of this (that is to say no rows) will be what appears in the footer on the last page of the table. The marker from the first row is shown in Figure 15-33 and the marker from the last row is shown in Figure 15-34.

Figure 15-33: FO for marker in the first table row

```
<table-row>
 <marker marker-class-name="continued">
  <table-row>
   <table-cell>
    <block>continued on next page/<block>
   </table-cell>
  </table-row>
 </marker>

 <table-cell>
  <block>row 1 cell 1 /<block>
 </table-cell>
</table-row>
```

Figure 15-34: FO for marker in the last table row

```
<table-row>
 <marker marker-class-name="continued"/>
 <table-cell>
  <block>row (last) cell 1 /<block>
 </table-cell>
<table-row>
```

15.13 Aligning columns at the decimal point

Ibex can align the contents of cells in a column on the decimal point by specifying text-align="." on each fo:table-cell in the column. This can be done explicity on each fo:table-cell, or to make things easier to maintain it can be done by specifying text-align="." on the fo:table-column and text-align="from-table-column" on each fo:table-cell.

Example FO for aligning columns is shown in Figure 15-35 and the resulting output is shown in Figure 15-36.

Figure 15-35: FO for decimal point alignment

```
<table font="10pt arial">
  <table-column column-width="50%" />
  <table-column column-width="50%" text-align="."/>
  <table-body>
    <table-row>
      <table-cell border="1pt solid black" padding="3pt" >
        <block>ibexdls</block>
      </table-cell>
      <table-cell border="1pt solid black" padding="3pt"
         text-align="from-table-column()">
        <block>499.02</block>
      </table-cell>
    </table-row>
    <table-row>
      <table-cell border="1pt solid black" padding="3pt" >
        <block>Total</block>
      </table-cell>
      <table-cell border="1pt solid black" padding="3pt"
         text-align="from-table-column()" font-size="18pt">
        <block>499.00</block>
      </table-cell>
    </table-row>
  </table-body>
</table>
```

Figure 15-36: Output for decimal point alignment

ibexdls	499.02
Total	499.00

Chapter 16

Images

Images are added to the document using either the external-graphic or instream-foreign-object elements. The external-graphic element is used to include a file in JPEG, GIF, TIFF, BMP, SVG or PNG formats. The instream-foreign-object element is used to include an image defined in Scalable Vector Graphics (SVG) format where the image SVG is contained within the FO.

The properties used to format the external-graphic and instream-foreign-object elements are the same.

The size of the image is distinct from the size of the area in which the image is placed.

The height and width attributes on the external-graphic or instream-foreign-object element specify the size of the area into which the graphic will be placed. If these properties are not specified they default to an area large enough to contain the image.

The content-width and content-height attributes control the size of the image. These can be values such as "3cm" or percentages such as "120%". If content-width and content-height not specified the image defaults to the size in pixels specified in the image file itself. This means that if you do not specify any of the above attributes the image will be as large as specified in the image file, and will be placed in an area the same size.

The dots per inch (dpi) value of the image makes a difference to the image size. Two images can have the same dimensions in pixels but appear different sizes in the PDF file. This is because Ibex uses the dpi value to work out the size of the image. An image which is 300 pixels wide and stored at 300 dpi will be 1 inch wide. The same image stored at 100 dpi will be 3 inches wide.

An image is an inline element, so for formatting purposes it can be placed in a sentence surrounded by text and is treated as a single word.

16.1 Image basics

The external-graphic element is used to include an image which is in a file external to the FO file. The name of the file to be included is specified using the src attribute.

The src attribute is called a *uri-specification* and must follow the following rules:

> A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

This means the following are all valid values for the src attribute:

uri(ibex.jpg)

uri("ibex.jpg")

uri('ibex.jpg')

url(http://www.xmlpdf.com/images/download2.gif)

As the src attribute is a URL, an image which exists on a web server can be downloaded automatically by Ibex as the PDF file is created. This is common in real estate and catalog applications and means you do not need to make a copy of an existing image just to get it into the PDF file. The FO shown in Figure 16-1 will fetch the file download2.gif from www.xmlpdf.com. The resulting image is shown in Figure 16-2.

Figure 16-1: FO to insert an image from a web server

```
<block space-before="6pt">
   <external-graphic border="1pt solid black"
     src="url(http://www.xmlpdf.com/images/download2.gif)"
   content-width="200%" content-height="200%"/>
</block>
```

Figure 16-2: Image included from web server

The external-graphic element can be used to include image files in PNG, JPEG, TIFF, BMP and GIF formats. It can also be used to include SVG images held in external files.

The inline-foreign-object is used for loading images from SVG content that is contained inline in the FO. See SVG Images on page 107.

16.2 Making an image fit a specified space

To make an image fit a specified size use the height and width attributes to specify the size of the external-graphic element, and then use the content-width and content-height to fit the image to that size.

For example to fit an image into an area 2cm x 2cm, set the width and height attributes to "2cm" and set the content-width and content-height attributes to "scale-to-fit", as shown in Figure 16-3.

Figure 16-3: Scaling an image

```
<fo:external-graphic src="url(image.jpg)"
                     height="2in" width="2in"
                     content-height="scale-to-fit"
                     content-width="scale-to-fit"/>
```

If you only want the image reduced in size to fit the specified area and do not want it increased in size if it is smaller, specify content-width="scale-down-to-fit". This also applies to content-height.

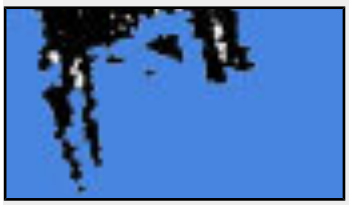
If you only want the image enlarged to fit the specified area and do not want it reduced in size if it is larger, specify content-width="scale-up-to-fit". This also applies to content-height.

16.3 Clipping

If the image is larger than the area in which it is contained then the image *may* be clipped. Figure 16-4 shows an image at its natural size, based on the pixels and dpi values read from the image file. If we specify the height of the external-graphic element as 2.5cm and specify overflow="hidden", the image will be clipped to this height, as shown in Figure 16-5.

Figure 16-4: Image at natural size

Figure 16-5: Clipped image

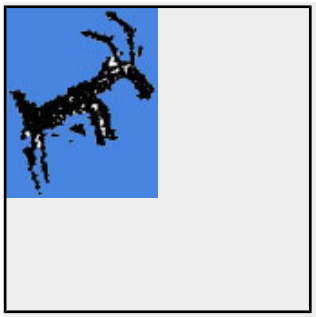


If we specify the height of the external-graphic element as 2.5cm and do not specify overflow="hidden", the image will not be clipped to this height, and will overwrite other content as shown to the right. Because the image is positioned on the same baseline as text, the overflow will be at the top of the area containing the image.

16.4 Image size and alignment

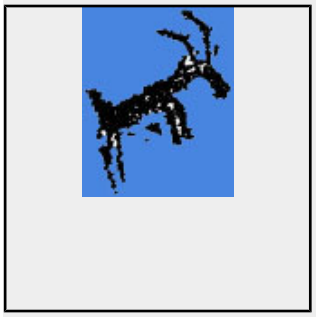
If an image is smaller than the containing area we can control where it appears in that area using the display-align and text-align attributes. The display-align attribute controls the vertical alignment, text-align controls the horizontal alignment. By default the image appears in the top left corner of the inline area created by the external-graphic or instream-foreign-object element, as shown in Figure 16-6.

Figure 16-6: Default alignment of an image



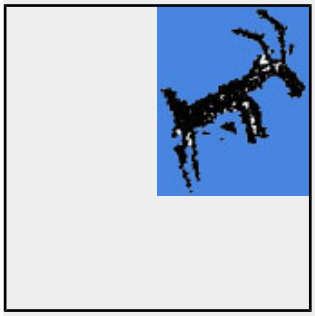
If we specify text-align="center" the image will move to the horizontal center of the inline area, as shown in Figure 16-7.

Figure 16-7: Using text-align = 'center'



If we specify text-align="right" the image will move to the right of the inline area as shown in Figure 16-8.

Figure 16-8: Right aligned image



If we specify text-align="center" and display-align="center" the image will move to the horizontal and vertical center of the inline area, as shown in Figure 16-9.

Figure 16-9: Vertically and horizontally centered image

16.4.1 Leading

Looking at the image in Figure 16-10 you can see a gap between the top of the image and the border. This is the leading, which appears because the image is treated as a text element and sits on the baseline. The amount of leading is derived from the font size, so you can reduce it to zero by setting the font size to zero, by specifying font-size="opt" on the containing block element. This has been done in Figure 16-11.

Figure 16-10: Image with leading above it

Figure 16-11: Using with leading removed

16.5 Image resolution

The resolution of an image in dots per inch (dpi) can be set using the dpi attribute on the external-graphic element. Setting this attribute overrides the dpi value read from the image file.

Setting the dpi to a lower value than the one specified in the image will result in smaller image of lower quality than the original. This is often done to reduce the size of the image in the PDF file and can result in massive decreases in PDF file size. If you have an image which is stored at 2400 dpi, and your end user will display it on a 96 dpi screen or print it on 600 dpi printer, reducing the image dpi to 600 will not effect the appearance of the image.

Setting the dpi to a value higher than the value read from the image file has no effect.

If for example if we wanted to store an image in the PDF file at 1200 dpi, we would use the FO shown in Figure 16-12.

Figure 16-12: FO to set image dpi

```
<block space-before="6pt">
  <external-graphic border="1pt solid black"
      src="url(http://www.xmlpdf.com/images/download2.gif)"
      content-width="200%" content-height="200%"
      dpi="1200"/>
</block>
```

The dpi attribute is an Ibex extension. It is not part of the XSL-FO standard.

16.6 Image anti-aliasing

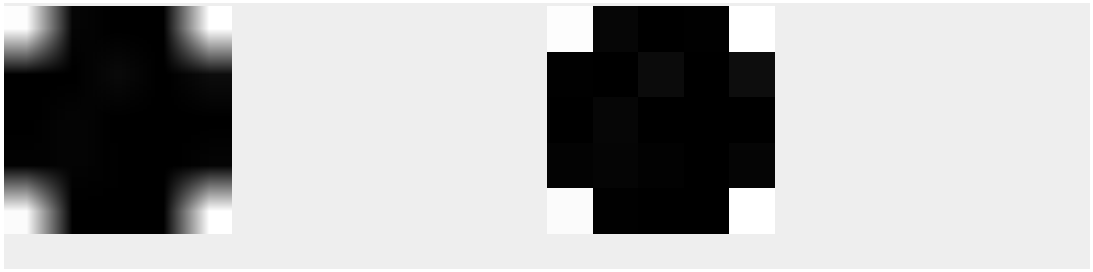
Images are anti-aliased by default. This can be disabled using the ibex:anti-alias attribute as shown in figure Figure 16-13.

Figure 16-13: FO to disable anti-aliasing

```
<block space-before="6pt">
  <external-graphic
      src="url(http://www.xmlpdf.com/images/download2.gif)"
      ibex:anti-alias="false"
      dpi="1200"/>
</block>
```

Figure 16-14 shows two images, the left right one has anti-aliasing disabled so the edges of the image appear more clearly.

Figure 16-14: Images with and without anti-aliasing



The ibex:anti-alias attribute is an Ibex extension. It is not part of the XSL-FO standard.

16.7 Loading an image from memory

Ibex has the facility to load an image which is stored in memory. This permits an application to dynamically generate an image or to load an image from a database for inclusion in the PDF file.

The image must be passed to Ibex in a byte array or a Stream (from the System.IO namespace).

The image must be given a unique identifier by which it can be retrieved during the PDF creation process. This is done using the addNamedImage() method on the FODocument object. This method takes two parameters; (1) a string which identifies the image and (2) the stream or array which contains the image itself.

For example if we had an image in a byte array called "image" and we wanted to give it the identifier "1029" we would use the code shown in Figure 16-15 to do this.

Figure 16-15: C# code to load an image from memory

```
byte[] image = ... dynamically create

FODocument document = new FODocument();

document.addNamedImage( "1029", image );
```

This must be done before calling generate() to create the PDF file.

Within the FO file the image is retrieved from memory using the syntax shown in Figure 16-16

Figure 16-16: FO to load an image from memory

```
<external-graphic src="url(data:application/ibex-image,1029)"/>
```

The value of the src attribute must be the string "url(data:application/ibex-image," followed by the unique identifier which was passed to addNamedImage().

This syntax for the url attribute conforms to RFC 2397 - The "data" URL scheme (which can be found at http://www.faqs.org/rfcs/rfc2397.html).

16.8 Transparent Images

16.8.1 Transparent GIF images

GIF images which have transparent areas are supported. The FO in Figure 16-17 places the same transparent GIF image on two different backgrounds. The output from this FO is shown in Figure 16-18.

Figure 16-17: FO for transparent image

```
<block background-color="blue">
   <external-graphic src="url(ibm-logo.gif)" content-height="2cm"/>
</block>
<block background-color="black">
   <external-graphic src="url(ibm-logo.gif)" content-height="2cm"/>
</block>
```

Figure 16-18: Transparent GIF images

16.8.2 Transparent PNG images

PNG images which have transparent areas are supported. The FO in Figure 16-19 places a transparent PNG image on a white backgrounds. The output from this FO is shown in Figure 16-20.

Figure 16-19: FO for transparent image

```
<block background-color="white">
   <external-graphic src="url(RedbrushAlpha-0.25.png)" content-height="2cm"/>
</block>
```

Figure 16-20: Transparent PNG image

16.8.3 Transparent images using masking

Ibex can use *image masking* to make some parts of an image appear transparent. This is an extension to the XSL-FO standard.

Image masking works by defining colors from the image which should not be displayed. The PDF viewer will compare each pixel in the image with the mask and not display

pixels which match the mask, effectively making these pixels transparent and so leaving visible the content behind the image.

The image mask is defined using the <ibex:mask> element, which must be contained within an external-graphic element, as shown in Figure 16-21.

Figure 16-21: FO to mask an image

```
<external-graphic src="url(ixsl.jpg)"  z-index='10'>
 <ibex:mask
        red-min="255" red-max="255"
        green-min="255" green-max="255"
        blue-min="255" blue-max="255"/>
</external-graphic>
```

To use the ibex:mask element you must reference the ibex namespace in your FO as shown in Figure 16-22.

Figure 16-22: Referencing the ibex namespace

```
<root
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">
```

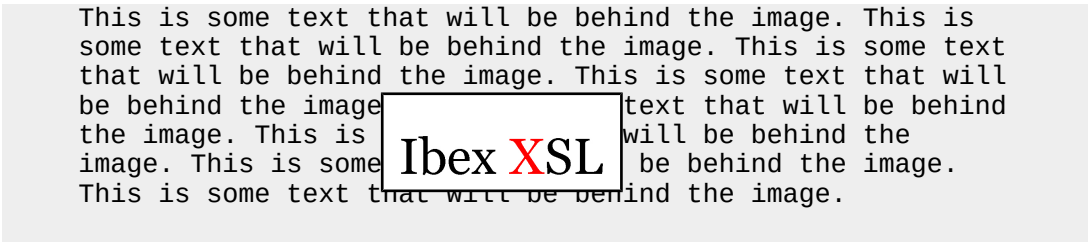
The mask defines the minimum and maximum values for each of the red, green and blue components of an image. A mask using these values is applicable only to images which are in RGB format with 24 bits per pixel.

For CMYK images, attributes called c-min, c-max, m-min, m-max, y-min, y-max, k-min and k-max define the minimum and maximum values for each of the cyan, magenta, yellow and black components of the image.

The image mask shown above causes any pixel which has red=255, green=255 and blue=255 to not be rendered. As a pixel with red, green and blue all equal to 255 is white, this means any white pixels will not be rendered.

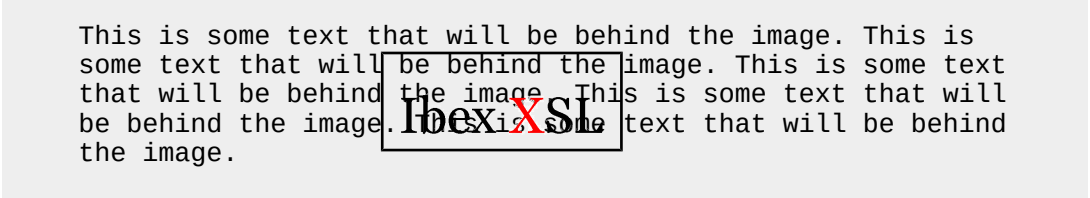
Figure 16-23 shows some text over which we have placed an image with red and black letters on a white background.

Figure 16-23: Image placed over text


This is some text that will be behind the image. This is
some text that will be behind the image. This is some text
that will be behind the image. This is some text that will
be behind the image
text that will be behind
the image. This is
will be behind the
image. This is some
be behind the image.
This is some text that will be behind the image.
Ibex XSL

If we add a mask to eliminate white pixels the image then appears as shown in Figure 16-24.

Figure 16-24: Image with masking


This is some text that will be behind the image. This is
some text that will be behind the image. This is some text
that will be behind the image. This is some text that will
be behind the image. This is some text that will be behind
the image.
Ibex XSL

16.8.4 Transparent Images using SVG

Transparent images can also be implemented by placing a SVG image over the top of other content. This approach uses the vector SVG renderer introduced in Ibex 2.1.2 and is

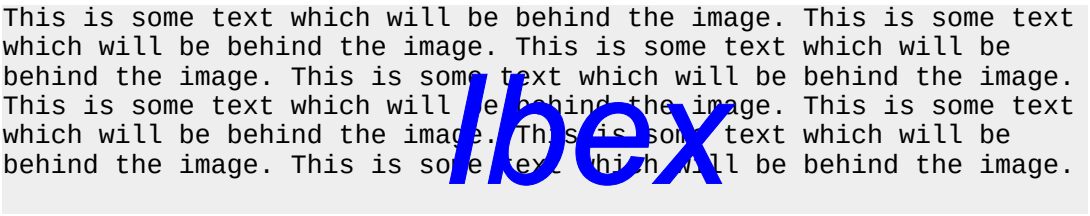
only available when using .NET 1.1 or higher. This is the best approach for transparent images because (a) there is no background on the SVG image so the best clarity is achieved, and (b) SVG uses a vector renderer which creates a smaller PDF file than you would get using a bitmap image.

Figure 16-25 shows the FO to put the word "ibex" over some text. The resulting output is shown in Figure 16-26.

Figure 16-25: FO using SVG to place content over text

```
<block-container>
  <block-container space-before="6pt"
    absolute-position="absolute" top="-1.6cm"
        left="5cm">
    <block>
       <instream-foreign-object z-index="30">
          <svg width="315" height="100"
              xmlns="http://www.w3.org/2000/svg">
           <text  x="30" y="60" fill="blue" stroke="blue"
              font-size="61pt" font-style="italic"
              style="font-family:arial;stroke-width:0.5">
              Ibex
           </text>
          </svg>
       </instream-foreign-object>
    </block>
  </block-container>
</block-container>
```

Figure 16-26: Text overlaid using SVG


This is some text which will be behind the image. This is some text
which will be behind the image. This is some text which will be
behind the image. This is some text which will be behind the image.
This is some text which will be behind the image. This is some text
which will be behind the image. This is some text which will be
behind the image. This is some text which will be behind the image.
Ibex

16.9 Network credentials used in accessing images

Ibex can retrieve images from HTTP servers as shown in Figure 16-27 below. By default Ibex will do this using the credentials of the process which is creating the PDF file. If Ibex is running in an ASP.NET server then the default configuration is that ASP runs as the ASPNET user. This user does not have permissions to access other servers and so will not be able to retrieve images from other servers.

Figure 16-27: FO to insert an image from an HTTP server

```
<block space-before="6pt">
   <external-graphic border="1pt solid black"
     src="url(http://www.xmlpdf.com/images/download2.gif)"
   content-width="200%" content-height="200%"/>
</block>
```

The FODocument object supports the setNetworkCredentials() method. This method takes 4 parameters, as shown in Figure 16-28.

Figure 16-28: The setNetworkCredentials method

```
public void setNetworkCredentials(
    string prefix,
    string username,
    string password,
    string domain )
```

The parameters are:

prefix — The start of a URL, such as "http://www.xmlpdf.com". Any image URL which starts with this prefix will use these credentials.

username — the username passed to the remote server

password — the password passed to the remote server

domain — the domain name passed to the remote server

Each call to setNetworkCredentials() is passed a prefix which is compared with image URLs to see which set of credentials to use.

For example if your application accesses two HTTP servers using different credentials your code might look like the code in Figure 16-29. Obviously you would get the username and password information from somewhere in your application rather than hard coding them.

Figure 16-29: Setting credentials for different servers

```
FODocument doc = new FODocument()

doc.setNetworkCredentials( "http://www.xmlpdf.com","user1","password1","domain1" );
doc.setNetworkCredentials( "http://www.ibex4.com","user2","password2","domain2" );
```

Internally Ibex uses the System.Net.WebRequest and System.Net.NetworkCredential objects to pass credentials to the remote server. If credentials have been passed to Ibex using the setNetworkCredentials() method a new NetworkCredential object is created when creating the WebRequest object. SO the actual forwarding of the credentials to the remote server is all done by the .NET framework.

Calls to setNetworkCredentials() should be made before the generate() method is called.

16.10 Multi-page TIFF image processing

Ibex has an extension attribute "ibex:page" which is used to specify which page of a multi-page TIFF image should be included in the PDF file.

FO to place the third page of a multi-page TIFF image is shown in Figure 16-30.

Figure 16-30: Specifying the page of a multi-page TIFF image

```
<block>
  <external-graphic src="url('7pages.tif')" ibex:page="3"/>
</block>
```

Chapter 17

Scalable Vector Graphics (SVG) images

Ibex supports the inclusion of Scalable Vector Graphics (SVG) images in the PDF file. SVG images retain their vector format inside the PDF file, so will remain precise under high magnification unlike bitmap images which will be come pixellated.

SVG images are inserted into the document using either the <fo:external-graphic> or <fo:instream-foreign-object> elements. Images can be included inline like this:

```
<fo:block border="1pt solid red">
  <fo:instream-foreign-object>
    <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20">
      <rect width="10" height="10" fill="green"/>
    </svg>
  </fo:instream-foreign-object>
</fo:block>
```

or from an external file like this:

```
<fo:block border="1pt solid red">
  <fo:external-graphic src="url(file.svg)"/>
</fo:block>
```

where the external file contains the SVG image like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20">
  <rect width="10" height="10" fill="green"/>
</svg>
```

If an image occurs more than once in the document it should be loaded from an external file so that it is only store in the PDF once.

17.2 Namespaces

The SVG image must begin with the <svg> element in the namespace "http://www.w3.org/2000/svg". Images which do not declare the namespace will not be included in the PDF.

17.3 Image size

The size of the image should be specified using the width and height attributes the outer <svg> element. These can be absolute measurements such as "3cm" or scalar values such as "400". Scalar values are assumed to be pixels and are converted to inches based on 1 pixel = 1/96 inch. Percentages cannot be effectively used; the size of the block containing the image is determined from the size of the image, at the time the image is processed the size of the containing block is unknown.

17.4 Summary of supported elements

This section briefly documents the degree to which SVG elements are supported in Ibex. It is not an SVG manual. Information on the SVG specification can be found at http://www.w3.org/TR/SVG11/expanded-toc.html

Animation of SVG elements using javascript is not supported.

17.4.1 <svg>

The <svg> element is used to define the size and shape of the image (using the width and height attributes) and to establish a new coordinate system using the viewBox attribute.

17.4.2 <g>

The <g> element used to move the coordinate system using the transform attribute. Supported transform operations are:

Operation	Effect
translate(x,y)	translate the coordinate system x units horizontally and y units vertically
translate(x)	translate the coordinate system x units horizontally and zero units vertically
matrix(a,b,c,d,e,f)	multiply the current transformation matrix by the one specified
scale(x,y)	scale the coordinate system x units horizontally and y units vertically
rotate(angle)	rotate the coordinate system angle degrees about the origin
rotate(angle,x,y)	rotate the coordinate system angle degrees about the point x,y
skewX(angle)	skew the coordinate system angle degrees along the X axis
skewY(angle)	skew the coordinate system angle degrees units along the Y axis

Multiple transformations can be performed by placing them one after the other in the transform attribute, like this:

```
<g transform="translate(10,20) scale(2,3) rotate(30)">
```

Transforms will be applied in the order in which they appear.

17.4.3 <defs>

The <defs> element is supported as a container for other elements. See <symbol> below for an example.

17.4.4 <desc>

The <desc> element is ignored.

17.4.5 <title>

The <title> element is ignored.

17.4.6 <symbol>

The <symbol> element is supported. The following image shows an example of defininig a system using <symbol> and retrieving it using <use>.

```
<?xml version="1.0" standalone="yes"?>
<svg width="10cm" height="3cm" viewBox="0 0 100 30"
      xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <symbol id="MySymbol" viewBox="0 0 20 20">
      <rect x="1" y="1" width="8" height="8"/>
      <rect x="11" y="1" width="8" height="8"/>
      <rect x="1" y="11" width="8" height="8"/>
      <rect x="11" y="11" width="8" height="8"/>
    </symbol>
  </defs>

  <use x="45" y="10" width="10" height="10" xlink:href="#MySymbol" fill="blue" />

</svg>
```

The <use> element will find the symbol element with id="#MySymbol" and display the content of this element, which should look like this:

17.4.7 <use>

The <use> element is supported, see above for an example. Note that as this element uses the xlink:href attribute it is necessary to declare the xmlns:xlink="http://www.w3.org/1999/xlink" namespace.

17.4.8 <image>

The <image> element is supported. This element embeds an image inside the SVG image. For example this image will display a rectangle and on top of that display the image held in the file "use_symbol.svg":

```
<?xml version="1.0"?>
<svg width="4cm" height="2cm" viewBox="0 0 200 100"
xmlns:xlink="http://www.w3.org/1999/xlink"
      xmlns="http://www.w3.org/2000/svg" version="1.1" preserveAspectRatio="none">

  <rect width="300" height="150" stroke="red" stroke-width="1" fill="silver"/>

  <image x="20" y="20" xlink:href="use_symbol.svg" width="100" height="100"/>

</svg>
```

17.4.9 <switch>

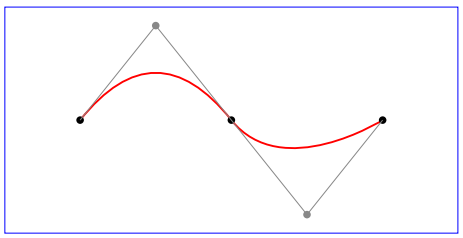
The <switch> element is ignored.

17.4.10 <path>

The <path> element is supported. Internally PDF does not support quadratic Bézier curves so they are converted to cubic Bézier curves. The following SVG draws a simple curve with marked end points:

```
<?xml version="1.0" standalone="no"?>
<svg width="6cm" height="5cm" viewBox="0 0 1200 600"
xmlns="http://www.w3.org/2000/svg">
  <rect x="1" y="1" width="1198" height="598" fill="none" stroke="blue"
stroke-width="1" />

  <path d="M200,300 Q400,50 600,300 T1000,300" fill="none" stroke="red"
stroke-width="5"  />
  <!-- End points -->
  <g fill="black" >
    <circle cx="200" cy="300" r="10"/>
    <circle cx="600" cy="300" r="10"/>
    <circle cx="1000" cy="300" r="10"/>
  </g>
  <!-- Control points and lines from end points to control points -->
  <g fill="#888888" >
    <circle cx="400" cy="50" r="10"/>
    <circle cx="800" cy="550" r="10"/>
  </g>
  <path d="M200,300 L400,50 L600,300
    L800,550 L1000,300"
        fill="none" stroke="#888888" stroke-width="2" />
</svg>
```

The curve looks like this:



17.4.10.1 Path line join shapes

The shape where a path changes direction is set with the stroke-linejoin attribute. Possible values are:

Value	Shape
stroke-linejoin="miter"	
stroke-linejoin="bevel"	
stroke-linejoin="round"	

17.4.11 <style>

The <style> element is currently implemented to some extent in .Net and not in Java. In .Net the class attribute can be used in conjunction with a style to apply that style to an element. Additional CSS support is currently being added to both Java and .Net.

17.4.12 <rect>

The <rect> element is supported. A simple rectangle can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="120" >
  <rect x="10" y="10" width="100" height="100" fill="none" stroke="red"/>
</svg>
```

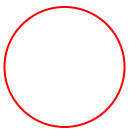
resulting in this image:

17.4.13 <circle>

The <circle> element is supported. A simple circle can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="120" >
    <circle cx="50" cy="50" r="30" fill="none" stroke="red"/>
</svg>
```

resulting in this image:

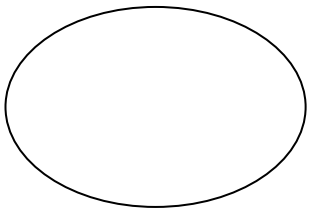


17.4.14 <ellipse>

The <ellipse> element is supported. A simple ellipse can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="200" >
  <ellipse cx="100" cy="100" rx="75" ry="50" fill="none" stroke="black"/>
</svg>
```

resulting in this image:



17.4.15 <line>

The <line> element is supported. A simple line can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="400" >
   <line x1="10" y1="10" x2="100" y2="10" stroke="blue" stroke-width="4"/>
</svg>
```

resulting in this image:

17.4.15.1 Line end shapes

The shape of the end of a line is set with the stroke-linecap attribute. Possible values are:

Value	Shape
stroke-linecap="butt"	
stroke-linecap="round"	
stroke-linecap="square"	The end of the line is the same shape as the default stroke-linecap="butt" but projects further beyond the end coordinate.

17.4.15.2 Dashed lines

Dashed lines are supported using the stroke-dasharray attribute. A dashed line can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="400" >
  <line x1="10" y1="10" x2="100" y2="10" stroke="blue" stroke-width="4"
stroke-dasharray="3 2"/>
</svg>
```

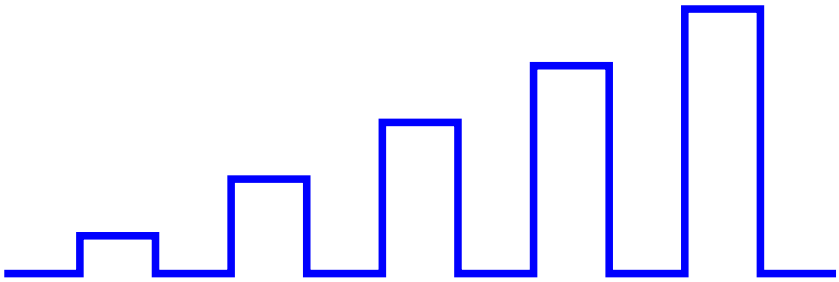
resulting in this image:

17.4.16 <polyline>

The <polyline> element is supported. A simple polyline can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="12cm" height="4cm"
  viewBox="0 0 1200 400">
  <polyline fill="none" stroke="blue" stroke-width="10"
            points="50,375
                    150,375 150,325 250,325 250,375
                    350,375 350,250 450,250 450,375
                    550,375 550,175 650,175 650,375
                    750,375 750,100 850,100 850,375
                    950,375 950,25 1050,25 1050,375
                    1150,375" />

</svg>
```

resulting in this image:

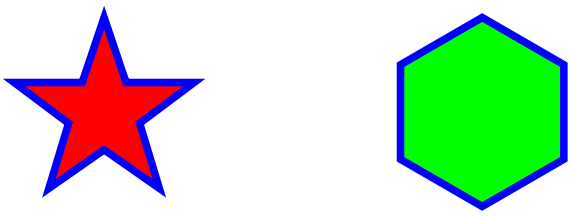


17.4.17 <polygon>

The <polygon> element is supported. A simple polygon can be drawn like this:

```
<svg xmlns="http://www.w3.org/2000/svg" width="12cm" height="4cm"
      viewBox="0 0 1200 400">
        <polygon fill="red" stroke="blue" stroke-width="10"
                    points="350,75  379,161 469,161 397,215
                    423,301 350,250 277,301 303,215
                    231,161 321,161" />
        <polygon fill="lime" stroke="blue" stroke-width="10"
                  points="850,75  958,137.5 958,262.5
                    850,325 742,262.6 742,137.5" />
</svg>
```

resulting in this image:



17.4.18 <text>

The <text> element is supported.

17.4.19 <tspan>

The <tspan> element is not implemented.

17.4.20 <textpath>

The <textpath> element is not implemented.

17.4.21 <pattern>

The <pattern> element is not implemented.

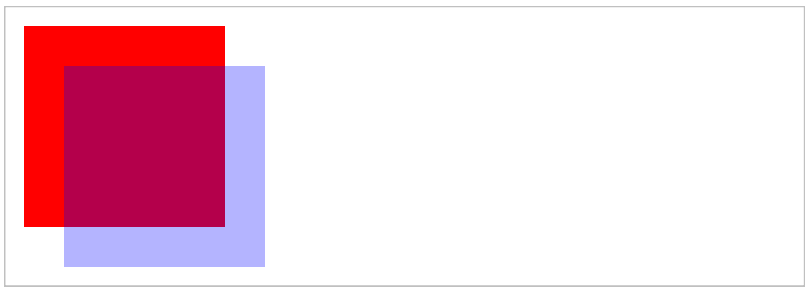
17.5 Opacity

The attributes stroke-opacity and fill-opacity are supported. Using the group opacity attribute to apply opacity to a group of elements is not supported, instead the opacity value is applied as if stroke-opacity and fill-opacity has been specified.

This example shows a transparent blue rectangle drawn over an opaque red rectangle.

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="140" >
  <rect width="400" height="140" fill="none" stroke="silver"/>
  <g transform="translate(10,10)">
    <rect width="100" height="100" fill="red"/>
  </g>
  <g transform="translate(30,30)">
    <rect width="100" height="100" fill="blue" stroke-width="1" fill-opacity="0.3" />
  </g>
</svg>
```

resulting in this image:

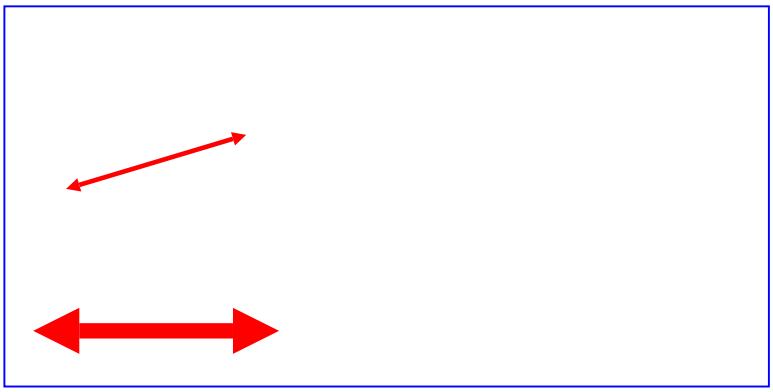


17.6 Markers

Markers are supported at the start and end of <line> and <path> elements. The <marker> element contains a separate drawing which can be reused. This example shows an arrowhead which is drawn at the each end of a line:

```
<?xml version="1.0" standalone="no"?>
<svg width="4in" height="2in"
     viewBox="0 0 4000 2000" version="1.1"
     xmlns="http://www.w3.org/2000/svg">
  <defs>
    <marker id="RedTriangle" viewBox="0 0 10 10" refX="0" refY="5"
      markerUnits="strokeWidth"
      markerWidth="4" markerHeight="3"
      orient="auto" fill="red">
      <path d="M 0 0 L 10 5 L 0 10 z" />
    </marker>
  </defs>
  <rect x="10" y="10" width="3980" height="1980"
       fill="none" stroke="blue" stroke-width="10" />

  <g transform="translate(400,1700) scale(.8)">
    <line x1="0" x2="1000" y1="0" y2="0" stroke="red" stroke-width="100"
marker-end="url(#RedTriangle)"
          marker-start="url(#RedTriangle)"/>
  </g>

  <g transform="translate(400,700) scale(.8)">
    <line x1="0" x2="1000" y1="300" y2="0" stroke="red" stroke-width="30"
marker-end="url(#RedTriangle)"
          marker-start="url(#RedTriangle)"/>
  </g>

</svg>
```

In this example the arrowhead appears once in the SVG, and is rendered four times. Each time it is rendered its rotaton and size are changed to match the rotation and size of the line.



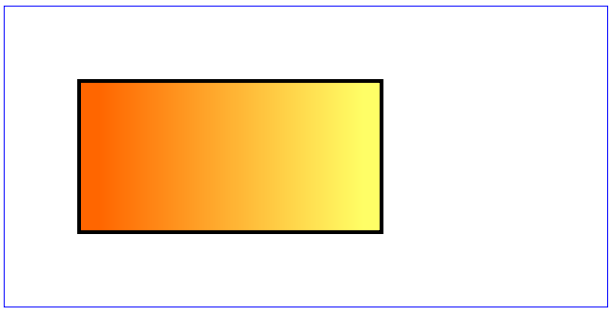
17.7 Linear gradients

Linear gradients are supported. This example produces a gradient from red to yellow horizontally:

```
<?xml version="1.0" standalone="no"?>
<svg width="8cm" height="4cm" viewBox="0 0 800 400" version="1.1"
      xmlns="http://www.w3.org/2000/svg">

  <g>
    <defs>
      <linearGradient id="MyGradient"
           x1="100" x2="500" gradientUnits="userSpaceOnUse">
        <stop offset="5%" stop-color="#F60" />
        <stop offset="95%" stop-color="#FF6" />
      </linearGradient>
    </defs>

    <rect fill="none" stroke="blue"
```

```
          x="1" y="1" width="798" height="398"/>

    <rect fill="url(#MyGradient)" stroke="black" stroke-width="5"
          x="100" y="100" width="600" height="200"/>
  </g>
</svg>
```

producing this image:



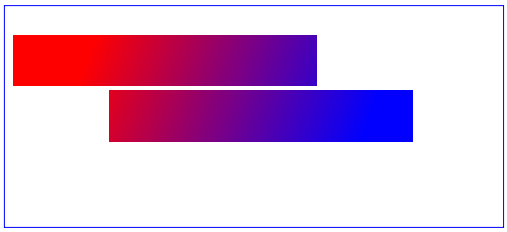
The interpretation of the values specified for the coordinates x1/x2/y1/y2 of the linearGradient element changes depending on value specified for gradientUnits.

When gradientUnits="userSpaceOnUse" the specified values are in "user space", which is the space defined by the prevailing <g> element. The specified coordinates are relative to the prevailing <g> element, so two elements which use the same gradient as their fill color will appear differently if they are placed in different locations on the page.

This SVG image shows rectangles using the same gradient in conjunction with gradientUnits="userSpaceOnUse"

```
<?xml version="1.0" standalone="no"?>
<svg width="8cm" height="3cm" viewBox="0 0 1000 450" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <g>
    <defs>
      <linearGradient id="linear_userSpaceOnUse" gradientUnits="userSpaceOnUse"
x1="100" y1="100" x2="700" y2="300">
          <stop offset="5%" stop-color="#ff0000" />
          <stop offset="95%" stop-color="#0000ff" />
      </linearGradient>
    </defs>

    <rect fill="none" stroke="blue" x="1" y="1" width="990" height="440"/>

    <g transform="translate(10,50)">
       <rect fill="url(#linear_userSpaceOnUse)" x="10" y="10" width="600"
height="100"/>
       <rect fill="url(#linear_userSpaceOnUse)" x="200" y="120" width="600"
height="100"/>
    </g>

  </g>
</svg>
```

producing this image:

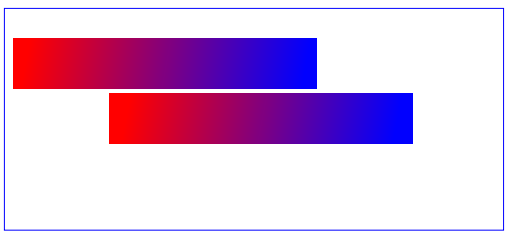


When gradientUnits="objectBoundingBox" the specified values are relative to the bounding box of the element being filled, and should be expressed as fractions of the dimensions of the element being filled. The values for coordinates should be in the range [0..1], so for example specifying x1="0" starts the gradient at the left hand edge of the element being filled, and specifying x1="0.2" starts the gradient at 20% of the width of that element. As the gradient is positioned relative to the element being filled, two element using the same gradient will appear the same regardless of the position of the element.

This SVG image shows rectangles using the same gradient in conjunction with gradientUnits="objectBoundingBox"

```
<?xml version="1.0" standalone="no"?>
<svg width="8cm" height="3cm" viewBox="0 0 1000 450" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <g>
    <defs>
      <linearGradient id="linear_objectBoundingBox" x1="0" y1="0" x2="1" y2="1">
        <stop offset="5%" stop-color="#ff0000" />
        <stop offset="95%" stop-color="#0000ff" />
      </linearGradient>

    </defs>

    <rect fill="none" stroke="blue" x="1" y="1" width="990" height="440"/>

    <g transform="translate(10,50)">
       <rect fill="url(#linear_userSpaceOnUse)" x="10" y="10" width="600"
height="100"/>
       <rect fill="url(#linear_userSpaceOnUse)" x="200" y="120" width="600"
height="100"/>
    </g>

  </g>
</svg>
```

producing this image:



17.8 Radial gradients

Radial gradients are supported from version 5.7.6 onwards.

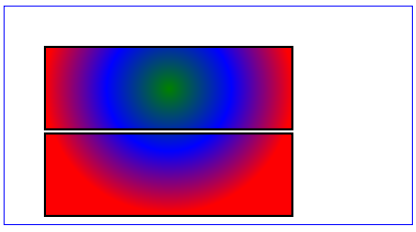
The interpretation of the values specified for the coordinates cx/cy/r/fx/fy of the radialGradient element changes depending on value specified for gradientUnits.

When gradientUnits="userSpaceOnUse" the specified values are in "user space", which is the space defined by the prevailing <g> element. The specified coordinates are relative to the prevailing <g> element, so two elements which use the same gradient as their fill color will appear differently if they are placed in different locations on the page.

This SVG image shows rectangles using the same gradient in conjunction with gradientUnits="userSpaceOnUse"

```
<?xml version="1.0" standalone="no"?>
<svg width="8cm" height="3cm" viewBox="0 0 1000 550" version="1.1"
xmlns="http://www.w3.org/2000/svg">

  <g>
    <defs>
      <radialGradient id="radial_userSpaceOnUse" gradientUnits="userSpaceOnUse"
cx="400" cy="200" r="300" fx="400" fy="200">
        <stop offset="0%" stop-color="green" />
        <stop offset="50%" stop-color="blue" />
        <stop offset="100%" stop-color="red" />
      </radialGradient>
    </defs>

    <rect fill="none" stroke="blue" x="1" y="1" width="990" height="530"/>
    <rect fill="url(#radial_userSpaceOnUse)" stroke="black" stroke-width="5" x="100"
y="100" width="600" height="200"/>
    <rect fill="url(#radial_userSpaceOnUse)" stroke="black" stroke-width="5" x="100"
y="310" width="600" height="200"/>
  </g>
</svg>
```

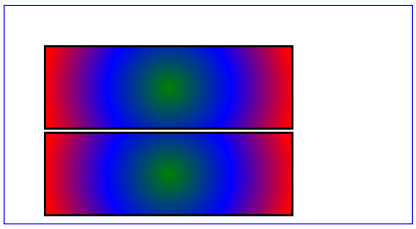
producing the image below, in which you can clearly see the gradient circles are centered within the first rectangle.



When gradientUnits="objectBoundingBox" the specified values are relative to the bounding box of the element being filled, and should be expressed as fractions of the dimensions of the element being filled. The values for coordinates should be in the range [0..1], so for example specifying x1="0" starts the gradient at the left hand edge of the element being filled, and specifying x1="0.2" starts the gradient at 20% of the width of that element. As the gradient is positioned relative to the element being filled, two element using the same gradient will appear the same regardless of the position of the element.

This SVG image shows rectangles using the same gradient in conjunction with gradientUnits="userSpaceOnUse"

producing the image below.



Chapter 18

Absolute Positioning

Content can be positioned anywhere on the page by placing the content in a block-container element and setting the absolute-position attribute.

If the absolute-position attribute is set to "fixed", the content will then be positioned on the page *relative to the page* area which contains the block-container element.

If the absolute-position attribute is set to "absolute", the content will be positioned on the page *relative* to the *reference area* which contains the block-container element. The reference area is not the containing block, it is the containing region, table-cell, block-container, inline-container or table-caption. In XSL-FO 1.0, the specification was ambiguous and the block-container was positioned relative to the containing area, but in XSL 1.1 this has been clarified to mean the containing reference area.

18.1 Positioning block-containers

It is important to realise that block-containers are not positioned relative to the containing block. Figure 18-1 shows FO with two absolutely positioned block containers. Both block-containers will be positioned relative to the containing region, because the region is the containing reference area. As they both have the same top attribute they will both be positioned in the same place.

Figure 18-1: Badly positioned block containers

```
<flow flow-name="body">
  <block>
    some text
    <block-container absolute-position="absolute" height="2cm" top="3cm">
       <block>
         in block-container one
       </block>
    </block-container>
  </block>

  <block>
    some more text
    <block-container absolute-position="absolute" height="2cm" top="3cm">
       <block>
         in block-container two
       </block>
    </block-container>
  </block>
</flow>
```

The simplest way to position a block-container is to place it inside another block-container which does not have the absolute-position attribute. FO for doing this is shown in Figure 18-2. The outer block-container is not absolutely positioned and will be

placed in the normal flow of content. The inner block-container is absolutely positioned relative to the outer one.

Figure 18-2: Positioned a block-container using another block-container

```
<flow flow-name="body">
  <block>
    some text
    <block-container>
      <block-container absolute-position="absolute" height="2cm" top="3cm">
         <block>
           in block-container one
         </block>
      </block-container>
    </block-container>
  </block>

  <block>
    some more text
    <block-container>
      <block-container absolute-position="absolute" height="2cm" top="3cm">
         <block>
           in block-container two
         </block>
      </block-container>
    </block-container>
  </block>
</flow>
```

18.2 Positioning and sizing block containers

A block-container with absolute-position = "absolute" is positioned relative to its containing reference area.

The distance between the left edge of the block-container and the left edge of the containing reference area is set by the left attribute. This attribute specifies the offset of the block-container's left edge from the containing reference area's left edge. The default value is "opt", which causes the two edges to be in the same place. Positive values of left move the left edge of the block-container to the right, making the block-container smaller.

The distance between the right edge of the block-container and the right edge of the containing reference area is set by the right attribute. This attribute specifies the offset of the block-container's right edge from the containing reference area's right edge. The default value is "opt", which causes the two edges to be in the same place. Positive values of right move the right edge of the block-container to the left, making the block-container smaller.

The distance between the top edge of the block-container and the top edge of the containing reference area is set by the top attribute. This attribute specifies the offset of the block-container's top edge from the containing reference area's top edge. The default value is "opt", which causes the two edges to be in the same place. Positive values of top move the top edge of the block-container downwards, making the block-container smaller.

The distance between the bottom edge of the block-container and the bottom edge of the containing reference area is set by the bottom attribute. This attribute specifies the offset of the block-container's bottom edge from the containing reference area's bottom edge. The default value is "opt", which causes the two edges to be in the same

place. Positive values of bottom move the bottom edge of the block-container upwards, making the block-container smaller.

If none of the left, right, top or bottom attributes is specified the block-container will be the same size as the reference area which contains it. This is because the offsets all default to "opt" so the edges of the block-container are the same as the edges of its containing reference area. This means a block-container with absolute-position= "absolute" which is placed in a region will by default fill that region.

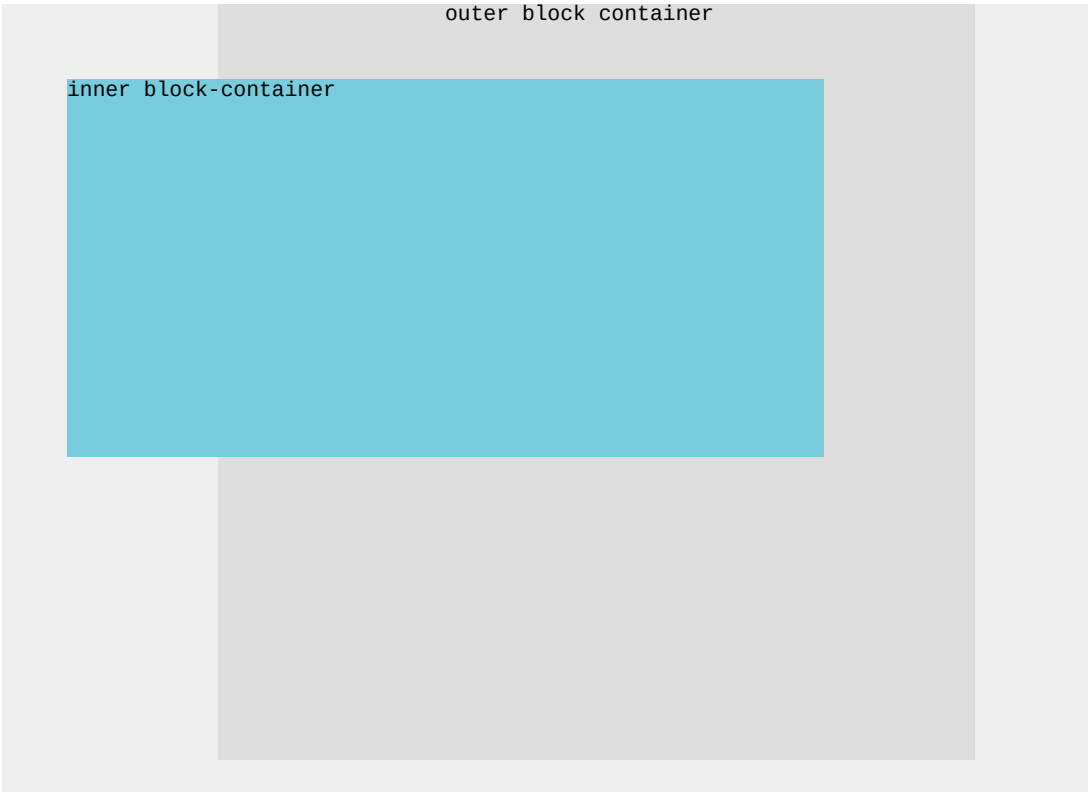
The height of a block-container can be specified with the height attribute, and the width with the width attribute.

Figure 18-3 shows the FO for a block container with height and width of 10cm, and an inner block-container which is offset from the outer one, including a negative offset on the left side. The output from this FO appears in Figure 18-4.

Figure 18-3: block-containers positioned and sized

```
<flow flow-name="body">
  <block>
    <block-container height="10cm" width="10cm" margin-left="3cm"
       background-color="#dddddd">
      <block>outer block container</block>
      <block-container absolute-position="absolute"
        top="1cm"
        right="2cm"
        left="-2cm"
        bottom="4cm"
        background-color="#77ccdd"
       >
         <block>
           inner block-container
         </block>
      </block-container>
    </block-container>
  </block>
</flow>
```

Figure 18-4: block-containers positioned and sized



The example in Figure 18-4 shows how using a negative left value will position the content to the left of the containing reference area. This is the technique used in this manual to place the labels next to each example.

Chapter 19

Columns

XSL-FO allows us to define a page which has multiple columns, just like this one.

This can only be done for whole pages, not for partial pages. However if we are in a region which has multiple columns we can treat it as a single-column region and place output across the whole width of the multi-column page by setting span="all" on block-level elements which appear immediately below the flow element.

Columns are defined by setting the column-count attribute of a body region element to a value greater than 1, and optionally setting the column-gap attribute to define a gap between the columns.

The page master for this is similar to the one shown in Figure 19-1.

Figure 19-1: The page master for a multi-column page

```
<simple-page-master
master-name="chapter-2col-odd">
    <region-start extent='2.5cm'/>
    <region-end extent='2.5cm'/>
    <region-body column-count='2'
       region-name="body"
margin='2.5cm'/>
    <region-after
region-name="footer-odd" extent="2.5cm"/>
    <region-before
region-name="header-odd" extent="2.5cm"/>
</simple-page-master>
```

All the blocks above, including this one, have span="all" set so that they span the whole page.

This block does not have span="all", so it will be fitted into the first column in the page. Text will flow to the bottom of this page and then start at the top of the next column.

If there are blocks above this one on the page which have span="all" (as there are) then they will remain in place and the text which is in only one column will be placed in the next column, below the span="all" blocks.

We deliberately repeat the paragraph to demonstrate this wrapping. This block does not have span="all", so it will be fitted into the first column in the page. Text will flow to the bottom of this page and then start at the top of the next column. If there are blocks above this one on the page (as there are) which have span="all" then they will remain in place and the text which is in only one column will be placed in the next column, below the span="all" blocks.

It is also possible to have a page start with content in two columns (like this).

When a block-level object is encountered which has span="all" the content already on the page is pushed up to the top, and the block with span="all" is spread over the two columns.

Chapter 20

Bookmarks

Bookmarks are the entries which appear on the right in a PDF file in Adobe Acrobat. They are used to navigate directly to locations within the document. They also have a hierarchical structure, where one bookmark can contain a set of child bookmarks which in turn can themselves contain other bookmarks.

The bookmark-tree element is optional. If used it should be placed under the root element, after the layout-master-set and declarations elements and before any page-sequence or page-sequence-wrapper elements.

The structure of a bookmark tree is shown in Figure 20-1.

Figure 20-1:
A bookmark tree

```
<bookmark-tree>
  <bookmark internal-destination="section-1">
    <bookmark-title>Chapter 1</bookmark-title>
    <bookmark internal-destination="section-1-1">
      <bookmark-title>Section 1</bookmark-title>
    </bookmark>
    <bookmark internal-destination="section-1-2">
      <bookmark-title>Section 2</bookmark-title>
    </bookmark>
  </bookmark>
  <bookmark internal-destination="section-2">
    <bookmark-title>Chapter 2</bookmark-title>
    <bookmark internal-destination="section-2-1">
      <bookmark-title>Section 1</bookmark-title>
    </bookmark>
  </bookmark>
</bookmark-tree>
```

We can see the following from the structure shown in Figure 20-1.

- The bookmarks are contained in a bookmark-tree element.
- A bookmark element has an internal-destination attribute identifying where in the document it links to. The value for this attribute should be used as the id attribute on the destination element.
- A bookmark element can contain other bookmark elements.
- The text which appears in the bookmark is contained within a bookmark-title element. Ibex supports using Unicode text in bookmarks.

The example above creates bookmarks like the ones in the Ibex manual.

The bookmarks which have child bookmark elements appear in the PDF file in a closed state, so the user can click the '+' next to them to display the child elements. If you specify starting-state="show" on a bookmark or bookmark-tree element it's immediate children will be visible when the PDF file is opened.

Chapter 21

Configuration

All configuration of Ibex is done using the Settings class which is accessed from the ibex4.FODocument object. This class has many properties which can be changed to configure the operation of Ibex.

Properties of the Settings class should be changed prior to calling the generate() method on the FODocument object. The fact that the Settings object is a property of the FODocument object means that different documents can have different Settings values. For example Figure 21-1 shows how to set the default line height to 1.4em.

Figure 21-1: Example usage of the Settings object

```
using System;
using ibex4;

public class IbexTester {

   public static void Main(string[] args) {

      FODocument doc = new FODocument()

      doc.Settings.LineHeightNormal = "1.4em";

      using( Stream xml =
             new FileStream( args[0], FileMode.Open, FileAccess.Read ) ) {
          using ( Stream output =
              new FileStream( args[1], FileMode.Create, FileAccess.Write ) ) {
                    doc.generate( xml, output );
               }
           }
   }
}
```

21.1 Fonts

The following properties on the Settings change the way fonts a processed. By default each absolute font size (small, medium, large etc.) is 1.2 times larger than the previous size.

Property	Type	Default	Notes
XX_Small	string	7.0pt	Must end in 'pt'.
X_Small	string	8.3pt	Must end in 'pt'.
Small	string	10.0pt	Must end in 'pt'.
Medium	string	12.0pt	Must end in 'pt'.
Large	string	14.4pt	Must end in 'pt'.
X_Large	string	17.4pt	Must end in 'pt'.
XX_Large	string	20.7pt	Must end in 'pt'.

Property	Type	Default	Notes
Smaller	string	0.8em	Must end in 'em'.
Larger	string	1.2em	Must end in 'em'.

21.2 Line height

The following properties on the Settings change the default line height. Ideally Settings.LineHeightNormal should end in 'em' to make line height proportional to the font height.

Property	Type	Default	Notes
LineHeightNormal	string	1.2em	

21.3 Page size

The following properties on the Settings change the default page size.

Property	Type	Default	Notes
PageHeight	string	297mm	
PageWidth	string	210mm	

21.4 Include paths

The following properties on the Settings effect retrieving XML or XSL files.

Property	Type	Default	Notes
BaseURI_XML	string		This value sets the base URI for including images and other XML files. When an external-graphic element specifies a relative path, Settings.BaseURI_XML is the base URI used in accordance with the rfc2396 URI Specification. When an XML file uses an entity to include another XML file, Settings.BaseURI_XML is the base URI used when Ibex searches for the included XML file.
BaseURI_XSL	string		This value sets the base URI for including other XSL files. When an xsl:include element is used to include another XSL stylesheet, Settings.BaseURI_XSL can be used to specify the location the included stylesheet should be loaded from.

21.5 Images

The following properties on the Settings effect retrieving images specified using the external-graphic element.

Property	Type	Default	Notes
BaseURI_XML	string		This value sets the base URI for including images and other XML files. When an external-graphic element specifies a relative path, Settings.BaseURI_XML is the base URI used in accordance with the rfc2396 URI Specification. When an XML file uses an entity to include another XML file, Settings.BaseURI_XML is the base URI used when Ibex searches for the included XML file.
WebRequestTimeoutMs	int	300	When an external-graphic element specifies an image is retrieved from a web server, this is the timeout used for the call to the web server. Units are milliseconds.

21.6 Border widths

The following properties on the Settings change the values for border widths specified with constants like 'thin'.

Property	Type	Default	Notes
BorderWidthThin	string	1pt	
BorderWidthMedium	string	2pt	
BorderWidthThick	string	3pt	

21.7 Layout

The following properties on the Settings change the appearance of content in the PDF file.

Property	Type	Default	Notes
OverflowIsVisible	bool	true	By default a region has overflow='auto', leaving it up the Ibex to decide whether content which overflows the bottom edge of a region is displayed or discarded. If Settings.OverflowIsVisible is true, the content will be displayed, if false it will be discarded. This property applies only if the XSL-FO attribute 'overflow' is not set or is set to 'auto'.

21.8 Leaders

The following properties on the Settings change the values for leader formatting objects.

Property	Type	Default	Notes
LeaderDot	char	.	When leader-pattern='dots', this is the character used as the dot

Chapter 22

Extensions

This chapter details Ibex-specific extensions to XSL-FO. Typically these extensions implement functionality such as password protecting a document which is not part of the XSL-FO standard.

The Ibex extensions have a namespace which is specified using the xmlns attribute as shown in Figure 22-1.

22.1 Document security

Ibex supports encryption of PDF documents and the setting of various document permissions. This is done using the ibex:security element as shown in Figure 22-1.

Figure 22-1: FO using the ibex:security element

```
<root xmlns="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:security deny-print='true' deny-extract='true'
 deny-modify='true' user-password='user' owner-password='owner'/>

...
```

Two levels of encryption are available, 40 bit and 128 bit. When using 40 bit encryption available permissions which can be set including deny-print, deny-extract and deny-modify. When using 128 bit encyption additional permissions can be set including deny-assembly and deny-print-high-resolution. These options are details in the sections below.

The level of encryption is specified using the bits attribute of the ibex:security element. This defaults to "40", so specify 128 bit encryption specify bits="128".

If used the ibex:security element **must** occur before any page-sequence elements.

22.1.1 40 bit encryption security options

When the number of bits of encryption is set to 40 or not specified, the attributes of the ibex:security element are:

Attribute	Values	Meaning
user-password		Specifies a password required to open the document in Acrobat. Once the document is opened with the correct user password, access is limited to permissions given using the attributes below.
owner-password		Specifies a password required to get all rights to the document in Acrobat. Once the document is opened with the correct owner password the user has total control of the document.
deny-print	true false	If this is set to true a user who opens the document with the user password will not be able to print the document.
deny-extract	true false	If this is set to true a user who opens the document with the user password will not be able to use cut-and-paste functionality to copy part of the document.
deny-modify	true false	If this is set to true a user who opens the document with the user password will not be able to modify the document.

Setting any of the attributes listed above will cause Ibex to encrypt the document.

Specifying the user-password but not the owner-password will set the owner-password to the same value as the user-password. This means anyone who can open the document using the user password has complete control of the document.

Specifying the owner-password but not the user-password is common usage. This means the user can open the document with limited rights without needing a password, but cannot then change or exceed those rights without knowing the owner password.

22.1.2 128 bit encryption security options

When the number of bits of encryption is set to 128, the attributes of the ibex:security element are:

Attribute	Values	Meaning
user-password		Specifies a password required to open the document in Acrobat. Once the document is opened with the correct user password, access is limited to permissions given using the attributes below.
owner-password		Specifies a password required to get all rights to the document in Acrobat. Once the document is opened with the correct owner password the user has total control of the document.
deny-print	true false	If this is set to true a user who opens the document with the user password will not be able to print the document.
deny-print-high-resolution	true false	If this is set to true a user who opens the document with the user password will not be able to print a high resolution copy of the document. They will only be able to print a low resolution (150dpi) version. If deny-print="true" this attribute has no effect and the document cannot be printed.
deny-extract	true false	If this is set to true a user who opens the document with the user password will not be able to use cut-and-paste functionality to copy part of the document.
deny-modify	true false	If this is set to true a user who opens the document with the user password will not be able to modify the document but can still "assemble" it. See deny-assembly below.
deny-assembly	true false	If deny-modify="true" and deny-assembly="false" then the user cannot change the document, but can "assemble" it, which means insert, rotate or delete pages and create bookmarks or thumbnail images. Setting deny-modify="true" and deny-assembly="true" prevents assembly.

Setting any of the attributes listed above will cause Ibex to encrypt the document.

Specifying the user-password but not the owner-password will set the owner-password to the same value as the user-password. This means anyone who can open the document using the user password has complete control of the document.

Specifying the owner-password but not the user-password is common usage. This means the user can open the document with limited rights without needing a password, but cannot then change or exceed those rights without knowing the owner password.

22.2 Standard document properties

Ibex allows you to set the various properties associated with a PDF document. These properties can be viewed in Acrobat by using the File | Document Properties | Summary menu option or just pressing control-d.

Figure 22-2 shows FO for setting the document properties using the ibex:properties element.

Figure 22-2: FO using ibex:properties

```
<root xmlns="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:properties
     title="Ibex User Manual"  subject="Ibex"
     author="visual programming limited"
     keywords="xml,pdf" creator="xtransform" />
...
```

If used the ibex:security element **must** occur before any page-sequence elements.

The attributes of the ibex:properties element are:

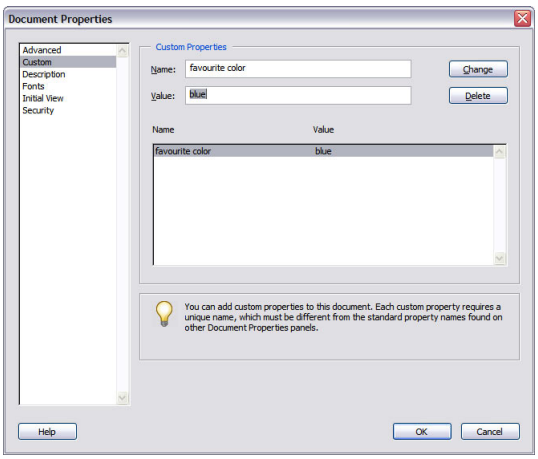
Attribute	Values	Meaning
title		Specifies a string which becomes the title property of the document.
subject		Specifies a string which becomes the subject property of the document.
author		Specifies a string which becomes the author property of the document.
keywords		Specifies a string which becomes the keywords property of the document. Separate individual keywords with commas.
creator		Specifies a string which becomes the creator property of the document. This should be the name of the application which created the XSL-FO document from which the PDF file was created.

Attribute	Values	Meaning
page-mode		Specifies how Acrobat will display the document when it is first opened. If set to 'bookmarks' then if the document has bookmarks they will be displayed. If set to 'thumbs' then the thumbnails tab in Acrobat will be displayed. If set to 'fullscreen' the document will be displayed without any toolbar, border etc.

Following the PDF standard, the document creator property should be the name of the product which converted the content to PDF format, so this is always Ibex. Other document properties such as creation and modification date are populated automatically by Ibex.

22.3 Custom Document Properties

Acrobat supports the display and editing of custom document properties. These properties are a set of name value pairs stored within the PDF file. In Acrobat 6.0 these properties can be viewed by using the File | Document Properties menu option and clicking on the "Custom" entry in the list box to display a screen like this:



These custom properties are inserted into the PDF using the ibex:custom element as shown in Figure 22-3.

Figure 22-3: FO using the ibex:custom element

```
<root xmlns="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:properties title="Ibex User Manual">
    <ibex:custom name="favourite color" value="blue"/>
</ibex:properties>
...
```

Each property must have a name and value attribute.

22.4 Image processing

22.4.1 Image resolution

Ibex adds the dpi attribute to the external-graphic element to permit managing the dots per inch resolution of images. See Image resolution on page 100.

22.4.2 Anti-aliasing

Ibex adds the ibex:anti-alias attribute to the external-graphic element to permit disabling anti-aliasing in order to achieve clearer images. See Image anti-aliasing on page 100.

22.4.3 Multi-page TIFF image processing

Ibex adds the ibex:page attribute to the external-graphic element to specify which page of a muti-page TIFF image should be included in the PDF file. See Multi-page TIFF images on page 105.

22.5 Bookmarks

XSL-FO 1.0 had no support for creating bookmarks in the PDF file. XSL 1.1 now has this feature so the ibex:bookmark element is no longer supported.

The XSL 1.1 implementation of bookmarks is described on page 129.

22.6 Document base URL

The PDF format supports setting a base URL for links created with a basic-link element. This base URL is prepended to the destination specified with an external-destination attribute if (and only if) the specified destination does not start with a '/' character.

Figure 22-4 shows FO which creates a document with "http://www.xmlpdf.com" as the base URL and a link to the page "index.html". When the user clicks on the link in the PDF file, it will go to "http://www.xmlpdf.com/index.html".

Figure 22-4: FO setting the document base URL

```
<ibex:document-base-url value="http://www.xmlpdf.com"/>
..
<block>
   <basic-link external-destination='url(index.html)'>
      index.html
   </basic-link>
</block>
```

The base URL is a document-wide property and can be set only once.

This property should not be confused with the Settings.BaseURI value which specifies a base URI to be used when Ibex retrieves images, stylesheets and XML during creation of the PDF file.

22.7 Ibex version

The ibex:version element inserts the version number of Ibex used to create the PDF file. This is an inline element which inserts characters into the document. Figure 22-5 shows FO which uses this element and the output appears in Figure 22-6.

Figure 22-5: FO using ibex:version

```
<block xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">
  created with Ibex version <ibex:version/>
</block>
```

Figure 22-6: Output from ibex:version

```
created with Ibex version 4.7.7.6
```

22.8 PDF/X

Ibex can create PDF files which comply with the PDF/X standard. This is described in detail on page 143.

22.9 Viewer Preferences

Ibex can set flags on the PDF file which control how the viewer application, such as Acrobat Reader, will display the PDF file.

These flags are set using the ibex:viewer-preferences element as shown in Figure 22-7.

Figure 22-7: FO using the ibex:viewer-preferences element

```
<root xmlns="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:viewer-preferences hide-toolbar="true"/>
...
```

The attributes for the ibex:viewer-preferences element are:

Attribute	Values	Meaning
hide-toolbar	true false	Set to true to hide the viewer application's tool bars
hide-menubar	true false	Set to true to hide the viewer application's menu bar
hide-window-ui	true false	Set to true to hide the UI and just display the document content
fit-window	true false	Set to true to resize the viewer window to fit the document page size

Attribute	Values	Meaning
center-window	true false	Set to true to center the viewer window on the screen
display-doc-title	true false	Set to true to have the viewer display the document title in the viewer frame rather than the file name. The document title is set using the title attribute of the ibex:properties element as detailed on page 138.

Chapter 23

PDF/X

This chapter details Ibex-specific extensions to the XSL-FO XML to support creation of PDF files which conform to the PDF/X standard.

Ibex implements the PDF/X standard using the ibex:pfdx element as shown in Figure 23-1.

Figure 23-1: PDF/X

```
<root xmlns="http://www.w3.org/1999/XSL/Format"
 xmlns:ibex="http://www.xmlpdf.com/2003/ibex/Format">

<ibex:pdfx color-profile-name="WideGamutRGB.icc"/>

...
```

The Ibex extensions have a namespace which is specified using the xmlns attribute as shown above.

The ibex:pdfx element **must** occur before any output is generated.

Using the ibex:pdfx element will automatically set the document color space to CMYK.

The existence of the ibex:pdfx element causes Ibex to create a PDF/X compatible PDF file. The field Settings.PDFXMode used in earlier releases has been removed.

The attributes of the ibex:pdfx element are:

Attribute	Values	Meaning
color-profile-file-name		Full or relative path to a ICC color profile file
registry-name		Registry Name used in the PDF OutputIntents structure. If not specified this defaults to "http://www.color.org".
info		Optional text which will become the Info value in the first OutputIntents array entry.
output-condition-identifier		Optional text which will become the OutputConditionIdentifier value in the first OutputIntents array entry. This defaults to "Custom"

Attribute	Values	Meaning
output-condition		Optional text which will become the OutputCondition value in the first OutputIntents array entry. This defaults to "Custom". Acrobat proposes values such as "TR001 SWOP/CGATS".

The color profiles is read from the specified ICC file, compressed, and embedded in the PDF file.

23.1 Media box

The MediaBox size within the PDF file will be set to the size of the page as specified on the simple-page-master for that page.

23.2 Bleed box

The BleedBox size defaults to the MediaBox size. The BleedBox can be specified as a change from the MediaBox by specifying the ibex:bleed-width attribute on the simple-page-master. This attribute specifies the distance by which the BleedBox is smaller than the MediaBox as shown in Figure 23-2.

Figure 23-2: Setting the bleed box size

```
<simple-page-master page-height="313mm" page-width="226mm"
     master-name="page" ibex:bleed-width="3mm">
```

If one value is used it applies to all sides of the page, if two values are used the top and bottom edges use the first value and the left and right edges use the second. If there are three values the top is set to the first value, the sides are set to the second value, and the bottom is set to the third value. If there are four values, they apply to the top, right, bottom and left edges in that order.

The following attributes can be specified to set each side explicitly: bleed-top-width, bleed-bottom-width, bleed-right-width, bleed-left-width.

23.3 Trim box

The TrimBox size defaults to the BleedBox size. The TrimBox can be specified as a change from the BleedBox by specifying the ibex:trim-width attribute on the simple-page-master. This attribute specifies the distance by which the TrimBox is smaller than the BleedBox as shown in Figure 23-3.

Figure 23-3: Setting the trim box size

```
<simple-page-master page-height="313mm" page-width="226mm"
    master-name="page" ibex:trim-width="3mm">
```

If one value is used it applies to all sides of the page, if two values are used the top and bottom edges use the first value and the left and right edges use the second. If there are three values the top is set to the first value, the sides are set to the second value, and the bottom is set to the third value. If there are four values, they apply to the top, right, bottom and left edges in that order.

The following attributes can be specified to set each side explicitly: trim-top-width, trim-bottom-width, trim-right-width, trim-left-width.

23.4 Overprint

Overprint mode can be enabled for the entire page by specifying the ibex:ibex-overprint-stroking, ibex:overprint-nonstroking and ibex:overprint-mode attributes as shown in Figure 23-4.

Figure 23-4: Setting the overprint mode

```
<simple-page-master page-height="313mm" page-width="226mm"
  master-name="page" ibex:overprint-stroking="true"
  ibex:overprint-nonstroking="true" ibex:overprint-mode="1">
```

Chapter 24

Using Ibex with Visual Basic and ASP

Ibex ships with binaries and source code for a COM wrapper which enables Ibex to be used from Visual Basic 6.0 and ASP applications. This code can either be used as-is or as the basis for developing your own wrapper. This chapter describes how this COM wrapper works.

24.1 The COM Wrapper

Within the Ibex distribution is a subdirectory called "ibexcom". In this directory is the source for a .NET project. This source is used to build the file ibexcom.dll which is the COM wrapper for Ibex.

24.1.1 Building the Wrapper

The COM wrapper can be built using the "nmake" command. The makefile executes the commands show in Figure 24-1 to build and register ibexcom.dll and ibexcom.tlb.

Figure 24-1: Commands to build the wrapper

```
copy ..\ibex11.dll ibex11.dll
csc /r:ibex11.dll /target:library /out:ibexcom.dll /unsafe assemblyinfo.cs
       wrapper.cs comstream.cs ibexinterface.cs
regasm ibexcom.dll /tlb
gacutil -i ibexcom.dll
gacutil -i ibex11.dll
```

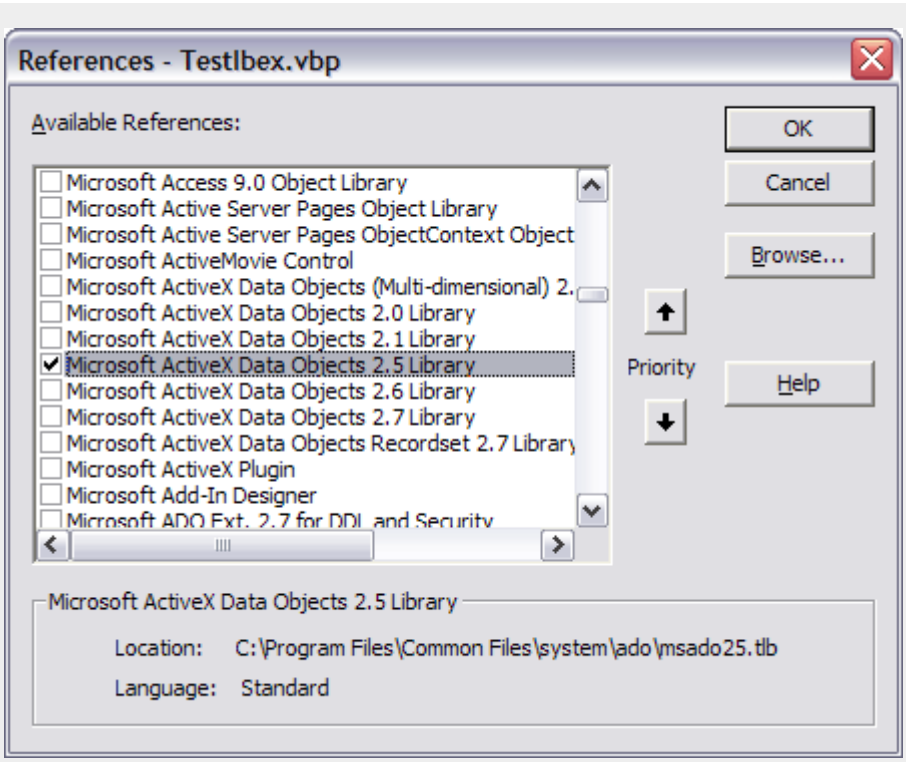
24.1.2 VB6 Example

This section shows how to use Ibex from VB6. Ibex ships with an example VB6 project in the file TestIbex.vbp. The steps below detail how this project was created. Before you can use the Ibex COM wrapper you should run the *nmake* command as described in the previous section.

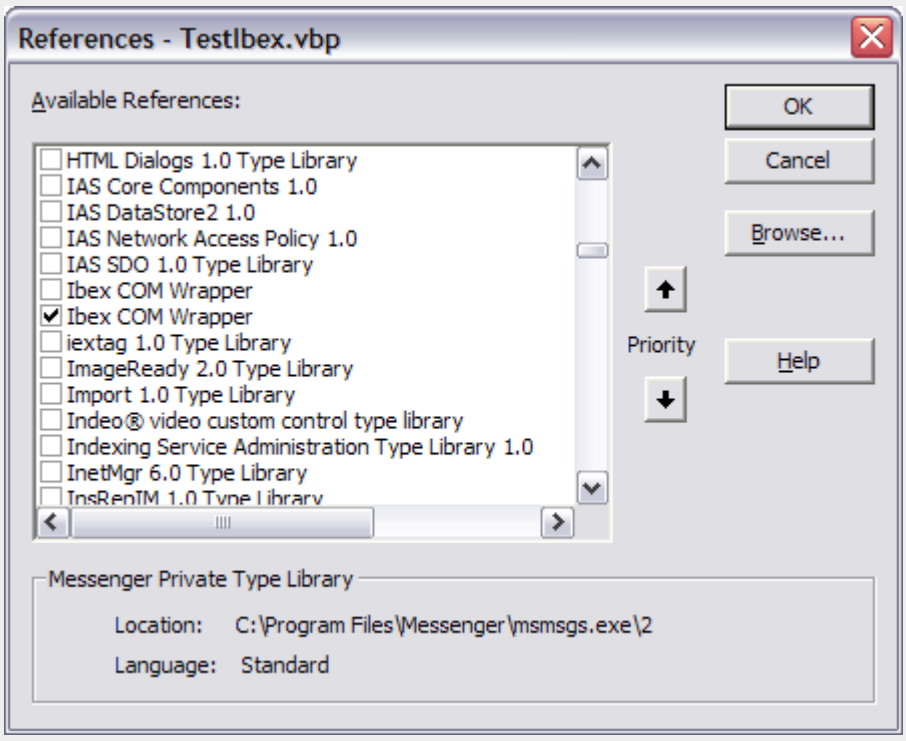
The XML files used in this example (book.xsl, hello.fo) ship with Ibex and are in the ibexcom directory.

First create a new VB 6 project, of type "Standard EXE". Use the Project->References menu option to add a reference to the both ADO 2.5 and ibexcom.tlb file. The dialog should look something like the one in Figure 24-2.

Figure 24-2:
Adding a reference to Ibex


References - TestIbex.vbp
Available References:
OK
Cancel
Browse...
Help
Priority
Microsoft Access 9.0 Object Library
Microsoft Active Server Pages Object Library
Microsoft Active Server Pages ObjectContext Object
Microsoft ActiveMovie Control
Microsoft ActiveX Data Objects (Multi-dimensional) 2.
Microsoft ActiveX Data Objects 2.0 Library
Microsoft ActiveX Data Objects 2.1 Library
Microsoft ActiveX Data Objects 2.5 Library
Microsoft ActiveX Data Objects 2.6 Library
Microsoft ActiveX Data Objects 2.7 Library
Microsoft ActiveX Data Objects Recordset 2.7 Library
Microsoft ActiveX Plugin
Microsoft Add-In Designer
Microsoft ADO Ext. 2.7 for DDL and Security
Microsoft ActiveX Data Objects 2.5 Library
Location: C:\Program Files\Common Files\system\ado\msado25.tlb
Language: Standard

and



The COM wrapper provides an interface called IbexComInterface, which can be used to declare a variable like this:

```
Dim ibex As ibexcom.IbexCOMInterface
```

The implementation of the interface is called Wrapper, and can be created like this:

```
    Set ibex = New ibexcom.Wrapper
```

Note that each instance of the wrapper can be used to create only one PDF document.

The code contained in the example ibextest.vbp project is:

```
 Private Sub Form_Load()

On Error GoTo errorhandler

    Dim ibex As ibexcom.IbexCOMInterface


    ' create the wrapper object
    Set ibex = New ibexcom.Wrapper

    ibex.SetLoggingLevelToInfo
    ibex.LogToFile "ibex.log"

    ' generate file to file
    ibex.GenerateFileFile "hello.fo", "hello.pdf"

    Set ibex = Nothing

    ' generate stream to stream
    Set ibex = New ibexcom.Wrapper

    Dim strmInput As New Stream

    strmInput.Open
    ' must match encoding of fo file or will get "invalid data at 1,1" message
    strmInput.Charset = "utf-8"
    strmInput.LoadFromFile "hello.fo"
    strmInput.Position = 0

    Kill "hellostream.pdf"

    Dim strmPDF As New Stream
    strmPDF.Open

    ibex.GenerateXMLStreamPDFStream strmInput, strmPDF, True

    Set strmInput = Nothing

    ' save the PDF stream to file, just to show it works
    strmPDF.Position = 0

    ' will get write error here if file exists
    strmPDF.SaveToFile "hellostream.pdf"
    Set strmPDF = Nothing

    Set ibex = Nothing

    ' test xslt translation and pdf creation
    Set ibex = New ibexcom.Wrapper

    Dim strmXML As New Stream

    strmXML.Open
    ' must match encoding of fo file or will get "invalid data at 1,1" message
    strmXML.Charset = "utf-8"
    strmXML.LoadFromFile "book.xml"
    strmXML.Position = 0

    Dim strmXSL As New Stream

    strmXSL.Open
    ' must match encoding of fo file or will get "invalid data at 1,1" message
    strmXSL.Charset = "utf-8"
    strmXSL.LoadFromFile "book.xsl"
    strmXSL.Position = 0
```

```
    Set strmPDF = New Stream
    strmPDF.Open

    ibex.GenerateXMLStreamXSLStreamPDFStream strmXML, strmXSL, strmPDF, True

    Set strmXML = Nothing
    Set strmXSL = Nothing

    ' save the PDF stream to file, just to show it works
    strmPDF.Position = 0
    strmPDF.SaveToFile "helloxsl.pdf"
    Set strmPDF = Nothing

    Set ibex = Nothing


    End

errorhandler:
    MsgBox Err.Description
    End

End Sub
```

Key things to note are:

A PDF file can be created from an FO file to a PDF file on disk using this call:

```
    ibex.GenerateFileFile "hello.fo", "hello.pdf"
```

A PDF file can be created from FO contained in Stream to a PDF Stream using this call:

```
    ibex.GenerateXMLStreamPDFStream strmInput, strmPDF, True
```

The final parameter (True in this case) indicates the output stream should be closed after the PDF file is created.

The input Stream object is part of the ADO 2.5 namespace, which is why we added a reference to ADO 2.5 earlier on. A stream can be populated from a file on disk using code like this:

```
    strmInput.Open

    ' must match encoding of fo file or will get "invalid data at 1,1" message
    strmInput.Charset = "utf-8"

    strmInput.LoadFromFile "hello.fo"
    strmInput.Position = 0
```

To create a PDF file using XSLT translation, use this call, with the XML and XSL read from Stream objects and the PDF written to a Stream:

```
      ibex.GenerateXMLStreamXSLStreamPDFStream strmXML, strmXSL, strmPDF, True
```

Chapter 25

Elements and Attributes

This chapter describes each major formatting object and its usage.

25.1 Declarations and pagination and layout formatting objects

The objects described in this section are used to define the geometry of the page and to control which content appears where on the page.

25.1.1 root

Description

This is the top level element in the FO and contains the layout-master-set, an optional declarations and one or more page-sequence elements. These child elements must be in the order listed.

Child element(s)

This element can contain the following elements:

bookmark-tree (zero or one)

declarations (zero or one)

layout-master-set (exactly one)

page-sequence (zero or more)

page-sequence-wrapper (zero or more)

Attributes

The following attributes can be used on this element:

media-usage	id
index-key	index-class

For an example showing the use of the element see Figure 25-1.

Figure 25-1: Using root

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </simple-page-master>
   </layout-master-set>

   <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
</root>
```

25.1.2 declarations

Description

The declarations formatting object is used to group global declarations for a stylesheet. In Ibex it acts as a container for the color-profile element which is used in PDF/X files. See 143 for more information.

Child element(s)

This element can contain the following elements:

color-profile

Parent element(s)

This element can be contained in the following elements:

root

25.1.3 color-profile

Description

This element is used to specify an external color profile file used in the creation of PDF/X files.

See 143 for more information.

Parent element(s)

This element can be contained in the following elements:

declarations

Attributes

The following attributes can be used on this element:

src color-profile-name

rendering-intent

25.1.4 page-sequence

Description

This element contains content for one or more pages. The content is contained in static-content elements which hold content for the page header, footer and other regions, and a one or more flow elements which contains content to be placed in the body regions of the page.

The page-sequence has a master-reference attribute which should correspond to the master-name of an element contained within the documents layout-master-set, such as a single-page-master.

The page number for the first page created by this page sequence can be set using the initial-page-number attribute. The format of the page number is controlled using the format attribute.

Child element(s)

This element can contain the following elements:

flow (one or more)

folio-prefix (zero or more)

folio-suffix (zero or more)

static-content (zero or more)

title (zero or more)

Parent element(s)

This element can be contained in the following elements:

root

page-sequence-wrapper

Attributes

The following attributes can be used on this element:

country	flow-map-reference
format	language
letter-value	grouping-separator
grouping-size	id
index-class	index-key
initial-page-number	force-page-count
master-reference	reference-orientation

writing-mode

For an example showing the use of the element see Figure 25-1.

25.1.5 page-sequence-wrapper

Description

This element is used to specify attributes which can be inherited by a group of page-sequence elements which are contained in the page-sequence-wrapper.

Child element(s)

This element can contain the following elements:

page-sequence (zero or more)

page-sequence-wrapper (zero or more)

Parent element(s)

This element can be contained in the following elements:

root

page-sequence-wrapper

Attributes

The following attributes can be used on this element:

id index-class

index-key

For an example showing the use of the element see Figure 25-2.

Figure 25-2: Using page-sequence-wrapper

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </simple-page-master>
   </layout-master-set>

   <page-sequence-wrapper index-key="main">
     <page-sequence master-reference="simple">
        <flow flow-name="body">
           <block>Hello World</block>
        </flow>
     </page-sequence>
   </page-sequence-wrapper>
</root>
```

25.1.6 layout-master-set

Description

This element contains all the page master elements (simple-page-master, page-sequence-master) used to create individual pages or sequence of pages.

At least one child element must exist or the document will contain no pages.

Child element(s)

This element can contain the following elements:

flow-map (zero or more)

page-sequence-master (zero or more)

simple-page-master (zero or more)

Parent element(s)

This element can be contained in the following elements:

root

For an example showing the use of the element see Figure 25-1.

25.1.7 page-sequence-master

Description

This element is used to define the sequence in which one or more page master elements (simple-page-master, repeatable-page-master) are used to create pages.

The element describes a sequence of page layouts and has a master-name which uniquely identifies it. This master-name is used as the master-reference on a page-sequence element in order to create pages using the sequence described by this page-sequence-master.

Each child of this element specifies a sequence of one or more pages:

A single-page-master-reference element is used define the layout for one page.

A repeatable-page-master-reference element is used define multiple pages which have the same layout because they use the same page master.

A repeatable-page-master-alternatives element is used define multiple pages which can have different layouts created using different page master elements.

Child element(s)

This element can contain the following elements:

repeatable-page-master-alternatives (zero or more)

repeatable-page-master-reference (zero or more)

single-page-master-reference (zero or more)

Parent element(s)

This element can be contained in the following elements:

layout-master-set

Attributes

The following attributes can be used on this element:

master-name

For an example showing the use of the element see Figure 25-3.

Figure 25-3: Using page-sequence-master

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"
            background-color='#eeeeee'/>
      </simple-page-master>
      <page-sequence-master master-name='repeated'>
          <repeatable-page-master-reference
             master-reference='simple'/>
      </page-sequence-master>
   </layout-master-set>

   <page-sequence master-reference="repeated">
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
</root>
```

25.1.8 single-page-master-reference

Description

This element specifies that the simple-page-master which has a master-name corresponding to the master-reference of this element should be used to define the layout for a single page.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

page-sequence-master

Attributes

The following attributes can be used on this element:

master-reference

For an example showing the use of the element see Figure 25-1.

25.1.9 repeatable-page-master-reference

Description

This element specifies that the simple-page-master which has a master-name corresponding to the master-reference of this element should be used to define the layout of one or more pages.

The difference between this and a single-page-master-reference is that the single-page-master-reference produces one page whereas this element can produce multiple pages. The maximum number of pages created by this element is controlled by the maximum-repeats attribute which by default is unlimited.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

page-sequence-master

Attributes

The following attributes can be used on this element:

master-reference maximum-repeats

For an example showing the use of the element see Figure 25-3.

25.1.10 repeatable-page-master-alternatives

Description

This element contains a set of conditional-page-master-reference elements, each of which specifies a page master and some conditional information.

When the rendering of content from a flow element triggers the creation of a new page each conditional-page-master-reference contained in this element is evaluated to see if it should be used.

Typically the conditional-page-master-reference elements are used to specify different page layouts for the first page of a sequence or for odd and even pages. The Ibex manual uses this approach, so that the first page of each chapter has no header.

Child element(s)

This element can contain the following elements:

conditional-page-master-reference (one or more)

Parent element(s)

This element can be contained in the following elements:

page-sequence-master

Attributes

The following attributes can be used on this element:

maximum-repeats

For an example showing the use of the element see Figure 25-4.

Figure 25-4: Using repeatable-page-master-alternatives

```
<page-sequence-master master-name='chapter'>
 <repeatable-page-master-alternatives>
   <conditional-page-master-reference
       page-position="first"
       master-reference='chapter-odd-no-header'/>

   <conditional-page-master-reference
        odd-or-even='odd'
        master-reference='chapter-odd'/>

   <conditional-page-master-reference
        odd-or-even='even'
        master-reference='chapter-even'/>
   </repeatable-page-master-alternatives>
</page-sequence-master>
```

25.1.11 conditional-page-master-reference

Description

This element associates a page master and a condition such that the page master will be used when the condition is true.

The conditions which are associated with this element are page-position, odd-or-even, and blank-or-not-blank. Each condition on each conditional-page-master-reference in a repeatable-page-master-alternatives element is evaluated in turn until one is found which is true, and that conditional-page-master-reference is used.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

repeatable-page-master-alternatives

Attributes

The following attributes can be used on this element:

master-reference	page-position
odd-or-even	blank-or-not-blank

For an example showing the use of the element see Figure 25-4.

25.1.12 simple-page-master

Description

This element defines the layout of a single page. It is uniquely identified by its master-name which is used on page-sequence and other elements to create pages which use this layout.

The content of the page goes into the named regions which are specified by the child elements of this element.

The size of the page is defined using the page-height and page-width attributes. The default page size is A4.

Child element(s)

This element can contain the following elements:

region-after (zero or one)

region-before (zero or one)

region-body (one or more)

region-end (zero or one)

region-start (zero or one)

Parent element(s)

This element can be contained in the following elements:

layout-master-set

Attributes

The following attributes can be used on this element:

margin	margin-bottom
margin-left	margin-right
margin-top	space-before
space-after	start-indent
end-indent	master-name
page-height	page-width
reference-orientation	writing-mode

For an example showing the use of the element see Figure 25-1.

25.1.13 region-body

Description

This element defines the shape of the main area on the page into which content from flow elements will be placed.

The region has a default name of "xsl-region-body" which is usually changed to something simpler using the region-name attribute.

A page can be defined which has multiple columns by using the column-count and column-gap attributes on this region.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

Ibex supports multiple body regions. There can be any number of body regions, provided each has a unique region-name. Content from different flows is mapped to different regions using the flow-map element.

The content of the region can be rotated using the reference-orientation attribute.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

simple-page-master

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top

border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
margin	margin-bottom
margin-left	margin-right
margin-top	space-before
space-after	start-indent
end-indent	clip
column-count	column-gap
display-align	overflow
region-name	reference-orientation
writing-mode	

For an example showing the use of the element see Figure 25-5.

Figure 25-5: Using regions

```
<simple-page-master master-name="front-page" margin='1.5cm'
     page-height="297mm" page-width="210mm">
  <region-body region-name="body"
         margin='0.75cm 0.5cm 0.75cm 3cm'/>
  <region-before region-name="header" extent="2.5cm"/>
  <region-after region-name="footer" extent="1cm"/>
  <region-start extent='1cm' background-color='#eeeeee'/>
  <region-end extent='1cm' background-color='#eeeeee'/>
</simple-page-master>
```

25.1.14 region-before

Description

This element defines the shape of a region which is at the top of a non-rotated page. Content from static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-before" which is usually changed to something simpler such as "header" using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the region-body element the region-before does not have margin properties. The size of the region is defined using the extent attribute.

By default the before region is reduced in width by the presence of the region-start and region-end elements. This can be changed by setting the precedence attribute to "true".

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

simple-page-master

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top

border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
clip	display-align
extent	overflow
precedence	region-name
reference-orientation	writing-mode

For an example showing the use of the element see Figure 25-5.

25.1.15 region-after

Description

This element defines the shape of a region which is at the bottom of a non-rotated page. Content from static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-after" which is usually changed to something simpler such as "footer" using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the region-body element the region-after does not have margin properties. The size of the region is defined using the extent attribute.

By default the before region is reduced in width by the presence of the region-start and region-end elements. This can be changed by setting the precedence attribute to "true".

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

simple-page-master

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top

border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
clip	display-align
extent	overflow
precedence	region-name
reference-orientation	writing-mode

For an example showing the use of the element see Figure 25-5.

25.1.16 region-start

Description

This element defines the shape of a region which is at the left of a non-rotated page. Content from static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-start" which is usually changed to something simpler such as "left" using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the region-body element the region-start does not have margin properties. The size of the region is defined using the extent attribute.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

simple-page-master

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style

border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
clip	display-align
extent	overflow
region-name	reference-orientation
writing-mode	

For an example showing the use of the element see Figure 25-5.

25.1.17 region-end

Description

This element defines the shape of a region which is at the right of a non-rotated page. Content from static-content elements whose flow-name matches the region-name will be placed in this region.

The region has a default name of "xsl-region-start" which is usually changed to something simpler such as "right" using the region-name attribute.

Within the region all of the content can be aligned to the top, bottom or middle of the region using the display-align attribute.

The content of the region can be rotated using the reference-orientation attribute.

Unlike the region-body element the region-end does not have margin properties. The size of the region is defined using the extent attribute.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

simple-page-master

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style

border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
clip	display-align
extent	overflow
region-name	reference-orientation
writing-mode	

For an example showing the use of the element see Figure 25-5.

25.1.18 flow

Description

This element contains block-level objects which create content which will appear in the body region of the page.

The flow-name attribute must correspond to a region-name used on the body region of the current page master for the content to be output. Which page master this is, is determined by the master-reference attribute of the containing page-sequence. If the flow-name does not match the region-name the content will not appear.

Child element(s)

This element can contain the following elements:

^%block (one or more)

Parent element(s)

This element can be contained in the following elements:

page-sequence

Attributes

The following attributes can be used on this element:

id	index-class
index-key	flow-name

25.1.19 static-content

Description

This element is used to create content in a region other then the body region. The term "static" refers to the fact that the content will go only on the current page, unlike the content of a flow element that may extend to many pages.

Static content is commonly used for page headers and footers. The content is usually different on each page as the page number changes.

The flow-name attribute may correspond to a region-name used on a non-body region of the current page master. Which page master this corresponds to is determined by the master-reference attribute of the containing page-sequence. If the flow-name does not match a region-name the content will not appear. This makes it possible to have a page-sequence which contains many static content elements each matching a different page layout. Only the static content which matches a region which is on the current page layout will be displayed.

Child element(s)

This element can contain the following elements:

block (zero or more)

block-container (zero or more)

float (one or more, cannot be used inside an out-of-line element)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

list-block (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

page-sequence

Attributes

The following attributes can be used on this element:

id	index-class

index-key flow-name

For an example showing the use of the element see .

25.1.20 title

Description

The title element associates a string title with a page sequence. This has no function when generating PDF so is discarded by Ibex.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

page-sequence

25.1.21 flow-map

Description

The flow-map is used to specify the assignment of flows to regions. Using the flow-map the content from one or more flow elements can be assigned to appear in one or more regions. So content from one flow can be rendered across multiple regions on the same page, filling one region then another.

flow-map elements must have a unique flow-map-name value. This is referenced by the flow-map-reference attribute of a page-sequence to assign the content of that page-sequence using the named flow-map.

flow-maps were added to XSL-FO in version 1.1.

Child element(s)

This element can contain the following elements:

flow-assignment (zero or more)

Parent element(s)

This element can be contained in the following elements:

layout-master-set

Attributes

The following attributes can be used on this element:

flow-map-name

For an example showing the use of the element see .

25.1.22 flow-assignment

Description

The flow-assignment is used to assign a list of flows to a list of regions.

Child element(s)

This element can contain the following elements:

flow-source-list (exactly one)

flow-target-list (exactly one)

Parent element(s)

This element can be contained in the following elements:

flow-map

For an example showing the use of the element see .

25.1.23 flow-source-list

Description

The flow-source-list contains a list of flow-name-specifier elements which specify the names of flows which will be assigned by the containing flow-map.

Child element(s)

This element can contain the following elements:

flow-name-specifier (one or more)

Parent element(s)

This element can be contained in the following elements:

flow-assignment

For an example showing the use of the element see .

25.1.24 flow-name-specifier

Description

The flow-name-specifier has one attribute which specifies the name of a flow. This is used in the flow-map element to add the named flow to a list of flows mapped to regions.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

flow-source-list

Attributes

The following attributes can be used on this element:

flow-map-reference

For an example showing the use of the element see .

25.1.25 flow-target-list

Description

The flow-target-list contains a list of region-name-specifier elements which specify the names of regions which will be assigned by the containing flow-map.

Child element(s)

This element can contain the following elements:

region-name-specifier (one or more)

Parent element(s)

This element can be contained in the following elements:

flow-assignment

For an example showing the use of the element see .

25.1.26 region-name-specifier

Description

The region-name-specifier has one attribute which specifies the name of a region. This is used in the flow-map element to add the named region to a list of regions mapped to flows.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

flow-target-list

Attributes

The following attributes can be used on this element:

region-name-reference

For an example showing the use of the element see .

25.2 Block level formatting objects

The objects described in this section are used to contain text and other block-level and inline-level elements.

25.2.1 block

Description

This element is the main container for text content. The simplest block element looks like this:

Figure 25-6:

```
<block>this is text</block>
```

The block is a block-level element. The other block-level elements are table, table-and-caption, list-block, and block-container.

A block element can contain other block-level elements as well as text. A typical usage would be to insert an empty block into a paragraph of text to cause a line break, like this:

Figure 25-7:

```
<block>this will be line 1</block>
<block/>
<block>this will be line 2</block>
```

Another use of nested blocks is to keep two other block-level objects together by using the keep-together attribute on the previous block, like this:

Figure 25-8:

```
<block keep-together="always">
     <block>this will be line 1</block>
     <block>this will be line 2</block>
</block>
```

To keep a block together and prevent it being split by a page break use the keep-together attribute.

To keep a block with the block following it use the keep-with-next attribute.

To keep a block with the block before it use the keep-with-previous attribute.

To format a block of text retaining line-feeds which were in the XML, use the linefeed-treatment attribute.

To change the color of text use the color attribute.

To align a paragraph to the left, right or both margins use the text-align and text-align-last attributes.

A block may contain a retrieve-marker only if the block is inside a static-content element.

Child element(s)

This element can contain the following elements:

basic-link (zero or more)

bidi-override (zero or more)

block (zero or more)

block-container (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

list-block (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

PCDATA

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

wrapper

basic-link

float

footnote-body

static-content

table-caption

block

block-container

inline

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
font-family	font-selection-strategy

font-size	font-stretch
font-size-adjust	font-style
font-variant	font-weight
margin	margin-top
margin-bottom	margin-right
margin-left	space-before
space-after	start-indent
end-indent	relative-position
bottom	top
right	left
break-after	break-before
color	text-depth
text-altitude	id
index-class	index-key
intrusion-displace	keep-together
keep-with-next	keep-with-previous
last-line-end-indent	linefeed-treatment
line-height	line-height-shift-adjustment
orphans	white-space-treatment
span	text-align
text-align-last	text-indent
visibility	white-space-collapse
widows	wrap-option

For an example showing the use of the element see Figure 25-1.

25.2.2 block-container

Description

This element is used to create an area (a "reference area" in the specifications terms) that has a different writing direction or rotation. If you want to achieve other ends such as keeping two blocks together use a block as the container.

If you do use reference-orientation to rotate the content to be vertical on the page then you need to specify inline-progression-dimension to limit the vertical height of the content.

The block-container element can be used to position content in a location relative to the page or to another block-container by setting the absolute-position attribute. See 123 for more information.

Child element(s)

This element can contain the following elements:

- block (zero or more)
- block-container (zero or more)
- float (one or more, cannot be used inside an out-of-line element)
- index-range-begin (one or more, subject to constraints specified for this element)
- index-range-end (one or more, subject to constraints specified for this element)
- list-block (zero or more)
- retrieve-marker (one or more, subject to constraints specified for this element)
- table (zero or more)
- table-and-caption (zero or more)
- wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- wrapper
- basic-link
- float
- footnote-body
- static-content
- table-caption
- block
- block-container

inline

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

absolute-position

top

left

background-color

background-repeat

background-position-vertical

border-before-color

border-before-width

border-after-style

border-start-color

border-start-width

border-end-style

border-top

border-top-style

border-bottom

border-bottom-style

border-left

border-left-style

border-right

border-right-style

padding

padding-after

padding-end

padding-bottom

bottom

right

background-attachment

background-image

background-position-horizontal

border

border-before-style

border-after-color

border-after-width

border-start-style

border-end-color

border-end-width

border-top-color

border-top-width

border-bottom-color

border-bottom-width

border-left-color

border-left-width

border-right-color

border-right-width

padding-before

padding-start

padding-top

padding-left

padding-right	margin
margin-top	margin-bottom
margin-right	margin-left
space-before	space-after
start-indent	end-indent
block-progression-dimension	break-after
break-before	clear
clip	display-align
height	id
index-class	index-key
inline-progression-dimension	intrusion-displace
keep-together	keep-with-next
keep-with-previous	overflow
reference-orientation	span
width	writing-mode
z-index	

For an example showing the use of the element see .

25.3 Inline level formatting objects

The objects described in this section are used directly contain and format text and other inline elements which are usually formatted across the page.

25.3.1 bidi-override

Description

This element is used when the Unicode BIDI algorithm fails to force some text to be written in a specific writing direction.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

block (zero or more)

block-container (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

list-block (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- block
- inline
- folio-suffix
- folio-prefix
- bidi-override
- marker

Attributes

The following attributes can be used on this element:

font-family	font-selection-strategy
font-size	font-stretch
font-size-adjust	font-style
font-variant	font-weight
margin	margin-top
margin-bottom	margin-right
margin-left	relative-position
bottom	top
right	left
color	direction
id	index-class
index-key	letter-spacing
line-height	score-spaces
unicode-bidi	word-spacing

25.3.2 character

Description

This element is used to insert a single character into the content. Given that modern XML editors can insert all Unicode characters there is little requirement to use this element.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width

border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
font-family	font-selection-strategy
font-size	font-stretch
font-size-adjust	font-style
font-variant	font-weight
margin	margin-top
margin-bottom	margin-right
margin-left	relative-position
bottom	top
right	left
alignment-adjust	treat-as-word-space
alignment-baseline	baseline-shift
character	color
dominant-baseline	text-depth
text-altitude	glyph-orientation-horizontal
glyph-orientation-vertical	id
index-class	index-key
keep-with-next	keep-with-previous

letter-spacing
line-height
score-spaces
suppress-at-line-break
text-decoration
text-shadow
text-transform
visibility
word-spacing

25.3.3 initial-property-set

Description

This element is used to format the first line of a block. It does not create any areas but its attributes are applied to the first line in the block which contains the initial-property-set.

Ibex does not currently implement this functionality

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

Attributes

The following attributes can be used on this element:

background-attachment
background-color
background-image
background-repeat
background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding

padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
font-family	font-selection-strategy
font-size	font-stretch
font-size-adjust	font-style
font-variant	font-weight
margin	margin-top
margin-bottom	margin-right
margin-left	relative-position
bottom	top
right	left
color	letter-spacing
line-height	score-spaces
text-decoration	text-shadow
text-transform	word-spacing

For an example showing the use of the element see .

25.3.4 external-graphic

Description

This element is used to include an image into the document.

This is an inline element so it must be contained in a block element.

The image source is defined by the src attribute.

The src attribute is called a *uri-specification* and must follow the following rules:

> A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or both be absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

This means the following are all valid values for the src attribute:

uri(ibex.jpg)

uri("ibex.jpg")

uri('ibex.jpg')

url(http://www.xmlpdf.com/images/download2.gif)

To set the size of the image use the content-height and content-width attributes.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
relative-position	bottom
top	right
left	margin

margin-top	margin-bottom
margin-right	margin-left
space-before	space-after
start-indent	end-indent
alignment-adjust	alignment-baseline
allowed-height-scale	allowed-width-scale
baseline-shift	block-progression-dimension
clip	content-type
content-height	content-width
display-align	dominant-baseline
height	id
index-class	index-key
inline-progression-dimension	keep-with-next
keep-with-previous	line-height
overflow	scaling
scaling-method	src
text-align	width

For an example showing the use of the element see .

25.3.5 instream-foreign-object

Description

This element is used to place an object which is contained in the XML into the PDF document. The only supported object type is an SVG image.

An example of include an inline SVG image is:

Figure 25-9:

```
<instream-foreign-object width="20%" height="1cm">
    <svg xmlns:fo="http://www.w3.org/TR/xsl/Format"
             xmlns="http://www.w3.org/2000/svg">
         <path style="stroke-width:1;fill:rgb(246,127,0);"
              d="M204.33 139.83 C196.33 133.33 z" />
    </svg>
</instream-foreign-object>
```

Not all implementations of Ibex support SVG images.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

- background-attachment
- background-color
- background-image
- background-repeat
- background-position-horizontal
- background-position-vertical
- border
- border-before-color
- border-before-style
- border-before-width
- border-after-color
- border-after-style
- border-after-width
- border-start-color
- border-start-style
- border-start-width
- border-end-color
- border-end-style
- border-end-width
- border-top
- border-top-color
- border-top-style
- border-top-width
- border-bottom
- border-bottom-color
- border-bottom-style
- border-bottom-width
- border-left
- border-left-color
- border-left-style
- border-left-width
- border-right
- border-right-color
- border-right-style
- border-right-width
- padding
- padding-before
- padding-after
- padding-start
- padding-end
- padding-top
- padding-bottom
- padding-left
- padding-right
- relative-position
- bottom
- top
- right
- left
- margin
- margin-top
- margin-bottom
- margin-right
- margin-left
- space-before
- space-after
- start-indent
- end-indent
- alignment-adjust
- alignment-baseline
- allowed-height-scale
- allowed-width-scale
- baseline-shift
- block-progression-dimension

clip
content-type
content-height
content-width
display-align
dominant-baseline
height
id
index-class
index-key
inline-progression-dimension
keep-with-next
keep-with-previous
line-height
overflow
scaling
scaling-method
text-align
width

25.3.6 inline

Description

This element is used to format some text in a way which is different to the containing block such as giving it a different font.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

block (zero or more)

block-container (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

list-block (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- footnote
- block
- inline
- folio-suffix
- folio-prefix
- bidi-override
- marker

Attributes

The following attributes can be used on this element:

- background-attachment
- background-color
- background-image
- background-repeat
- background-position-horizontal
- background-position-vertical
- border
- border-before-color
- border-before-style
- border-before-width
- border-after-color
- border-after-style
- border-after-width
- border-start-color
- border-start-style
- border-start-width
- border-end-color
- border-end-style
- border-end-width
- border-top
- border-top-color
- border-top-style
- border-top-width
- border-bottom
- border-bottom-color
- border-bottom-style
- border-bottom-width
- border-left

border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
font-family	font-selection-strategy
font-size	font-stretch
font-size-adjust	font-style
font-variant	font-weight
margin	margin-top
margin-bottom	margin-right
margin-left	relative-position
bottom	top
right	left
alignment-adjust	alignment-baseline
baseline-shift	block-progression-dimension
color	dominant-baseline
height	id
index-class	index-key
inline-progression-dimension	keep-together
keep-with-next	keep-with-previous
line-height	text-decoration
visibility	width
wrap-option	

For an example showing the use of the element see .

25.3.7 inline-container

Description

This element is used create an inline reference area. Because it can contain block-level elements it can be used to place a block-level element such as a table into a line of text. This can be used to horizontally center the block-level element by centering the inline-container, which being an inline element can be centered using normal text alignment attributes.

Child element(s)

This element can contain the following elements:

- block (zero or more)
- block-container (zero or more)
- float (one or more, cannot be used inside an out-of-line element)
- index-range-begin (one or more, subject to constraints specified for this element)
- index-range-end (one or more, subject to constraints specified for this element)
- list-block (zero or more)
- retrieve-marker (one or more, subject to constraints specified for this element)
- table (zero or more)
- table-and-caption (zero or more)
- wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- block
- inline
- folio-suffix
- folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment
background-color
background-image
background-repeat
background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
margin
margin-top
margin-bottom
margin-right
margin-left
relative-position
bottom
top
right
left
alignment-adjust
alignment-baseline
baseline-shift
block-progression-dimension

color
display-align
dominant-baseline
height
id
index-class
index-key
inline-progression-dimension
keep-together
keep-with-next
keep-with-previous
line-height
overflow
reference-orientation
width
writing-mode

25.3.8 leader

Description

This element is used to draw a horizontal line across the page.

A simple line is drawn like this:

Figure 25-10:

```
<block>
  <leader leader-pattern="rule" rule-thickness="0.2pt"/>
</block>
```

The leader can also be drawn between other pieces of text on the same line and can be set to expand to fill available space like this:

Figure 25-11:

```
<block text-align="justify" text-align-last="justify">
 This is before the leader
  <leader leader-pattern="rule" rule-thickness="0.2pt"/>
 this is after the leader
</block>
```

producing the effect below. Note the use of text-align-last which is required to justify the single line paragraph.

```
This is before the leader _____________________________ this is after the leader
```

Setting the leader-pattern attribute to "dots" changes the line into dots like this:

```
This is before the leader ............................. this is after the leader
```

Setting the leader-pattern attribute to "space" changes the line into spaces like this:

```
This is before the leader                                    this is after the leader
```

The use of leader-pattern = "use-content" is not supported.

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- block
- inline

folio-suffix

folio-prefix

bidi-override

marker

25.3.9 page-number

Description

This element is used to insert the current page number into the document.

The page-number string is formatted using the string conversion properties of the containing page-sequence, namely format, grouping-separator, grouping-size, letter-value, country and language.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color

border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
font-family	font-selection-strategy
font-size	font-stretch
font-size-adjust	font-style
font-variant	font-weight
margin	margin-top
margin-bottom	margin-right
margin-left	relative-position
bottom	top
right	left
alignment-adjust	alignment-baseline
baseline-shift	dominant-baseline
id	index-class
index-key	keep-with-next
keep-with-previous	letter-spacing
line-height	score-spaces
text-altitude	text-decoration

text-depth	text-shadow
text-transform	visibility
word-spacing	wrap-option

For an example showing the use of the element see .

25.3.10 page-number-citation

Description

This element is used to insert the first page number on which the content created by some other element occurs.

The page-number string is formatted using the string conversion properties of the containing page-sequence, namely format, grouping-separator, grouping-size, letter-value, country and language.

The page-number-citation has a ref-id attribute which should match the id attribute of the element whose page number we want to appear.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat

background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
font-family
font-selection-strategy
font-size
font-stretch
font-size-adjust
font-style
font-variant
font-weight
margin
margin-top
margin-bottom
margin-right
margin-left
relative-position
bottom
top
right
left
alignment-adjust
alignment-baseline
baseline-shift
dominant-baseline
id
index-class
index-key
keep-with-next
keep-with-previous
letter-spacing

line-height
ref-id
score-spaces
text-altitude
text-decoration
text-depth
text-shadow
text-transform
visibility
word-spacing
wrap-option

For an example showing the use of the element see Figure 25-12.

Figure 25-12: Example of page-number-citation

```
<?xml version='1.0' encoding='UTF-8'?>
<root xmlns="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"/>
      </simple-page-master>
   </layout-master-set>

  <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block id='22'>
         Hello
         </block>
      </flow>
  </page-sequence>

  <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>
           The block with id='22' starts on page
           <page-number-citation ref-id='22'/>
           and ends on page
           <page-number-citation-last ref-id='22'/>
         </block>
      </flow>
  </page-sequence>

</root>
```

25.3.11 page-number-citation-last

Description

This element is used to insert the last page number on which the content created by some other element occurs.

The page-number string is formatted using the string conversion properties of the containing page-sequence, namely format, grouping-separator, grouping-size, letter-value, country and language.

The page-number-citation-last has a ref-id attribute which should match the id attribute of the element whose page number we want to appear.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style

border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
font-family
font-selection-strategy
font-size
font-stretch
font-size-adjust
font-style
font-variant
font-weight
margin
margin-top
margin-bottom
margin-right
margin-left
relative-position
bottom
top
right
left
alignment-adjust
alignment-baseline
baseline-shift
dominant-baseline
id
index-class
index-key
keep-with-next
keep-with-previous
letter-spacing
line-height
ref-id
score-spaces
text-altitude
text-decoration
text-depth
text-shadow
text-transform

visibility word-spacing

wrap-option

For an example showing the use of the element see Figure 25-12.

25.3.12 folio-prefix

Description

This element is used create a prefix which appears before the page number (inserted by page-number, page-number-citation and page-number-citation-last).

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

page-sequence

25.3.13 folio-suffix

Description

This element is used create a suffix which appears after the page number (inserted by page-number, page-number-citation and page-number-citation-last).

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

page-sequence

25.3.14 scaling-value-citation

Description

This element is used to retrieve the amount by which an image was scaled when it was inserted into the PDF file.
The image should be identified with an id attribute which has the same value as the ref-id attribute on this element.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width

border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
font-family
font-selection-strategy
font-size
font-stretch
font-size-adjust
font-style
font-variant
font-weight
relative-position
bottom
top
right
left
alignment-adjust
alignment-baseline
country
dominant-baseline
format
grouping-separator
grouping-size
id
index-class
index-key
keep-with-next
keep-with-previous
language
letter-spacing
letter-value
line-height
intrinsic-scale-value
ref-id
score-spaces
scale-option
text-altitude

text-depth	text-shadow
text-transform	visibility
word-spacing	wrap-option

25.4 Formatting objects for tables

The objects described in this section are used to create tables.

25.4.1 table-and-caption

Description

This element is used to create a table which has a caption above or below it, and to keep the table and caption together.

By default the caption appears above the table. Set caption-side="bottom" to make the caption appear below the table.

Child element(s)

This element can contain the following elements:

table (zero or more)

table-caption (zero or one)

Parent element(s)

This element can be contained in the following elements:

wrapper

basic-link

float

footnote-body

static-content

table-caption

block

block-container

inline

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

background-attachment background-color

background-image
background-repeat
background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
margin
margin-top
margin-bottom
margin-right
margin-left
space-before
space-after
start-indent
end-indent
relative-position
bottom
top
right
left
break-after
break-before
caption-side
clear
id
index-class
index-key
intrusion-displace
keep-together
keep-with-next
keep-with-previous
text-align

25.4.2 table

Description

This element creates a table. Tables have rows and columns and possibly also headers and footers.

The size of table columns can either be calculated from the content of cells, or specified using table-column elements. Using table-column elements results in consistent output regardless of cell contents.

The width and other characteristics of columns are defined using table-column elements. An optional table header, which by default is repeated after each page break, is specified using the table-header element. An optional table footer, which by default is repeated before each page break, is specified using the table-footer element.

Table rows are contained in one or more table-body elements.

Table borders are controlled using the border-collapse attribute. If this has a value of "collapse" then table and cell borders are collapsed into a single border. If the value is "separate" then table, row and cell borders are all drawn separately, one inside the other.

The default value for border-collapse is "collapse". To create the kind of borders used in CSS where the cell borders appears inside the row and table borders set border-collapse to "separate".

Child element(s)

This element can contain the following elements:

table-body (one or more)

table-column (zero or more)

table-footer (zero or one)

table-header (zero or one)

Parent element(s)

This element can be contained in the following elements:

wrapper

basic-link

float

footnote-body

static-content

table-and-caption

table-caption

block

block-container

inline

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

background-attachment
background-color
background-image
background-repeat
background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right

font-family
font-selection-strategy
font-size
font-stretch
font-size-adjust
font-style
font-variant
font-weight
margin
margin-top
margin-bottom
margin-right
margin-left
space-before
space-after
start-indent
end-indent
relative-position
bottom
top
right
left
block-progression-dimension
border-after-precedence
border-before-precedence
border-collapse
border-end-precedence
border-separation
border-start-precedence
break-after
break-before
clear
id
index-class
index-key
inline-progression-dimension
intrusion-displace
height
keep-together
keep-with-next
keep-with-previous
table-layout
table-omit-header-at-break
table-omit-footer-at-break
width
writing-mode

25.4.3 table-column

Description

This element is used to specify characteristics for columns in a table such as the background color and the width.

A table would typically have multiple table-column elements looking something like this:

Figure 25-13:

```
<table>
    <table-column column-width="20%"/>
    <table-column column-width="30%"/>
    <table-column column-width="50%"/>
    <table-body>
         <table-row>
              <table-cell>col 1</table-cell>
              <table-cell>col 2</table-cell>
              <table-cell>col 3</table-cell>
         </table-row>
    </table-body>
</table>
```

This defines a table with three columns. Implicitly the three table-column elements specify the width of columns one, two and three in that order. This can be made explicit using the column-number attribute like this:

Figure 25-14:

```
<table>
    <table-column column-number="1"
      column-width="20%"/>
    <table-column column-number="2"
      column-width="30%"/>
    <table-column column-number="3"
      column-width="50%"/>
    <table-body>
         <table-row>
              <table-cell>col 1</table-cell>
              <table-cell>col 2</table-cell>
              <table-cell>col 3</table-cell>
         </table-row>
    </table-body>
</table>
```

A single table-column can be used to set the width and other characteristics of multiple columns by using the columns-spanned attribute. In the example below the first table-column sets the width of the first two columns to 20% and the third column to 50%:

Figure 25-15:

```
<table>
    <table-column columns-spanned="2"
         column-width="20%"/>
    <table-column
        column-width="50%"/>
    <table-body>
         <table-row>
              <table-cell>col 1</table-cell>
              <table-cell>col 2</table-cell>
              <table-cell>col 3</table-cell>
         </table-row>
    </table-body>
</table>
```

Percentage values used in the column-width attribute refer to the width of the table.

If table-column elements are not used all columns will be of equal width.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

table

Attributes

The following attributes can be used on this element:

background-attachment
background-color
background-image
background-repeat
background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom

padding-left
padding-right
border-after-precedence
border-before-precedence
border-end-precedence
border-start-precedence
column-number
column-width
number-columns-repeated
number-columns-spanned
visibility

25.4.4 table-caption

Description

This element is used to contain block-level formatting objects containing the caption for the table. It is used as part of a table-and-caption element.

Child element(s)

This element can contain the following elements:

block (zero or more)

block-container (zero or more)

float (one or more, cannot be used inside an out-of-line element)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

list-block (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

table-and-caption

Attributes

The following attributes can be used on this element:

background-attachment
background-color
background-image
background-repeat
background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style

border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
relative-position	bottom
top	right
left	block-progression-dimension
height	id
index-class	index-key
inline-progression-dimension	intrusion-displace
keep-together	width

25.4.5 table-header

Description

This element creates a header which appears once at the top of the table and is then repeated after each page break. To prevent this repetition set table-omit-header-at-break to "true" on the containing table.

A table-header is itself a table and contains rows and cells in the same manner as table element.

Child element(s)

This element can contain the following elements:

table-cell (zero or more)

table-row (zero or more)

Parent element(s)

This element can be contained in the following elements:

table

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style

border-left-width	border-right
border-right-color	border-right-style
border-right-width	margin
margin-top	margin-bottom
margin-right	margin-left
space-before	space-after
start-indent	end-indent
relative-position	bottom
top	right
left	border-after-precedence
border-before-precedence	border-start-precedence
border-end-precedence	id
index-class	index-key
visibility	

Notes on attributes

As described in section 6.7.6 of the XSL-FO specification, only the background properties from this set apply. If the value of border-collapse on the table is "collapse" or "collapse-with-precedence" the border properties also apply.

25.4.6 table-footer

Description

This element creates a footer which appears once at the bottom of the table and also before each page break. To prevent this repetition set table-omit-footer-at-break to "true" on the containing table.

A table-footer is itself a table and contains rows and cells in the same manner as table element.

Child element(s)

This element can contain the following elements:

table-cell (zero or more)

table-row (zero or more)

Parent element(s)

This element can be contained in the following elements:

table

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style

border-left-width	border-right
border-right-color	border-right-style
border-right-width	font-family
font-selection-strategy	font-size
font-stretch	font-size-adjust
font-style	font-variant
font-weight	margin
margin-top	margin-bottom
margin-right	margin-left
space-before	space-after
start-indent	end-indent
relative-position	bottom
top	right
left	border-after-precedence
border-before-precedence	border-start-precedence
border-end-precedence	id
index-class	index-key
visibility	

Notes on attributes

As described in section 6.7.7 of the XSL-FO specification, only the background properties from this set apply. If the value of border-collapse on the table is "collapse" or "collapse-with-precedence" the border properties also apply.

25.4.7 table-body

Description

This element is a container for table-row and table-cell elements. A single table element can contain multiple table-body elements which are output in the order in which they appear in the XML.

Child element(s)

This element can contain the following elements:

table-cell (zero or more)

table-row (zero or more)

Parent element(s)

This element can be contained in the following elements:

table

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style

border-right-width	border
border-before-color	border-before-style
border-before-width	border-after-color
border-after-style	border-after-width
border-start-color	border-start-style
border-start-width	border-end-color
border-end-style	border-end-width
border-top	border-top-color
border-top-style	border-top-width
border-bottom	border-bottom-color
border-bottom-style	border-bottom-width
border-left	border-left-color
border-left-style	border-left-width
border-right	border-right-color
border-right-style	border-right-width
relative-position	bottom
top	right
left	border-after-precedence
border-before-precedence	border-start-precedence
border-end-precedence	id
index-class	index-key
visibility	

Notes on attributes

As described in section 6.7.8 of the XSL-FO specification, only the background properties from this set apply. If the value of border-collapse on the table is "collapse" or "collapse-with-precedence" the border properties also apply.

25.4.8 table-row

Description

This element acts as a container for table-cell elements.

Table row elements are not required. A table-body element can contain table-cell elements directly using the starts-row and ends-row attributes on the cells to determine where rows start and end.

The height of a row is by default the height of the tallest cell in the row. This can be overridden using the height or block-progression-dimension attributes. Use block-progression-dimension.minimum to set a minimum height, block-progression-dimension.maximum to set a maximum height.

Rows cannot have padding. This is stated in section 6.7.9 of the XSL-FO specification.

Child element(s)

This element can contain the following elements:

table-cell (one or more)

Parent element(s)

This element can be contained in the following elements:

table-header

table-footer

table-body

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top

border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
relative-position
bottom
top
right
left
border-after-precedence
border-before-precedence
border-start-precedence
border-end-precedence
break-after
break-before
id
index-key
index-class
height
keep-together
keep-with-next
keep-with-previous
visibility

25.4.9 table-cell

Description

This element is a container for content in a cell within a table. Cell content is contained in block-level elements within the cell. A common error is to place text directly within the table-cell element, which results in the text being discarded.

A table-cell element can contain any number of block level elements.

Contents of a cell are aligned vertically using the display-align attribute.

To have a cell span mutiple columns use the number-columns-spanned attribute. To span multiple rows use the number-rows-spanned attribute.

Child element(s)

This element can contain the following elements:

- block (zero or more)
- block-container (zero or more)
- float (one or more, cannot be used inside an out-of-line element)
- index-range-begin (one or more, subject to constraints specified for this element)
- index-range-end (one or more, subject to constraints specified for this element)
- list-block (zero or more)
- retrieve-marker (one or more, subject to constraints specified for this element)
- table (zero or more)
- table-and-caption (zero or more)
- wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- table-header
- table-row
- table-footer
- table-body

Attributes

The following attributes can be used on this element:

- background-attachment
- background-color
- background-image
- background-repeat

background-position-horizontal
background-position-vertical
border
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top
border-top-color
border-top-style
border-top-width
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
padding
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
relative-position
bottom
top
right
left
border-after-precedence
border-before-precedence
border-start-precedence
border-end-precedence
block-progression-dimension
column-number
display-align
relative-align
empty-cells
ends-row
height
id
index-class
index-key
inline-progression-dimension
number-columns-spanned
number-rows-spanned
starts-row
width

25.5 Formatting objects for lists

The objects described in this section are used to create lists.

25.5.1 list-block

Description

This element is used to create a list, which is similar to a two column table.

A simple list looks like this:

Figure 25-16:

```
<list-block provisional-distance-between-starts=".5cm"
    provisional-label-separation="0.1cm">
   <list-item>
      <list-item-label end-indent="label-end()">
         <block font="8pt arial">&#x25CF;</block>
      </list-item-label>
      <list-item-body start-indent="body-start()">
           <block>
               item one
           </block>
       </list-item-body>
  </list-item>
   <list-item>
      <list-item-label end-indent="label-end()">
         <block font="8pt arial">&#x25CF;</block>
      </list-item-label>
      <list-item-body start-indent="body-start()">
           <block>
               item two
           </block>
       </list-item-body>
  </list-item>
</list-block>
```

producing the following content:

- item one
- item two

The list is rendered as two columns. The first column is called the label, the second is called the body.

The distance from the start of the label column to the start of the body column is set by the provisional-distance-between-starts attribute. The gap between the columns is set by the provisional-label-separation attribute. The width of the label column is therefore:

Figure 25-17:

```
provisional-distance-between-starts
  - provisional-label-separation
```

Each item in the list is contained in a list-item element. The list-item contains exactly one list-item-label and list-item-body element, with the list-item-label coming first.

The list-item-label should always have its end-indent attribute set to "label-end()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the end-indent is not so specified the label column will overlap the body column.

The list-item-body should always have its start-indent attribute set to "body-start()" which is a function returning a value calculated from the

provisional-distance-between-starts and provisional-label-separation attributes. If the start-indent is not so specified the label column will overlap the body column.

Child element(s)

This element can contain the following elements:

- list-item

Parent element(s)

This element can be contained in the following elements:

- wrapper
- basic-link
- float
- footnote-body
- static-content
- table-caption
- block
- block-container
- inline
- inline-container
- bidi-override
- table-cell
- list-item-label
- list-item-body
- marker

Attributes

The following attributes can be used on this element:

- background-attachment
- background-color
- background-image
- background-repeat
- background-position-horizontal
- background-position-vertical
- border
- border-before-color
- border-before-style
- border-before-width
- border-after-color
- border-after-style
- border-after-width
- border-start-color

border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
relative-position	bottom
top	right
left	margin
margin-top	margin-bottom
margin-right	margin-left
space-before	space-after
start-indent	end-indent
break-after	break-before
clear	id
index-class	index-key
intrusion-displace	keep-together
keep-with-next	keep-with-previous
provisional-distance-between-starts	provisional-label-separation

25.5.2 list-item

Description

This element contains the label and body of an entry in a list.

The height of the list-item will be the taller of the label and body items it contains.

Child element(s)

This element can contain the following elements:

list-item-body (exactly one)

list-item-label (exactly one)

Parent element(s)

This element can be contained in the following elements:

list-block

Attributes

The following attributes can be used on this element:

background-attachment	background-color
background-image	background-repeat
background-position-horizontal	background-position-vertical
border	border-before-color
border-before-style	border-before-width
border-after-color	border-after-style
border-after-width	border-start-color
border-start-style	border-start-width
border-end-color	border-end-style
border-end-width	border-top
border-top-color	border-top-style
border-top-width	border-bottom
border-bottom-color	border-bottom-style
border-bottom-width	border-left
border-left-color	border-left-style
border-left-width	border-right
border-right-color	border-right-style

border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
relative-position	bottom
top	right
left	margin
margin-top	margin-bottom
margin-right	margin-left
space-before	space-after
start-indent	end-indent
break-after	break-before
id	index-class
index-key	intrusion-displace
keep-together	keep-with-next
keep-with-previous	relative-align

For an example showing the use of the element see .

25.5.3 list-item-body

Description

This element contains the body part of a list item.

The list-item-body contains block-level elements, it does not itself contain text.

Child element(s)

This element can contain the following elements:

- block (zero or more)
- block-container (zero or more)
- float (one or more, cannot be used inside an out-of-line element)
- index-range-begin (one or more, subject to constraints specified for this element)
- index-range-end (one or more, subject to constraints specified for this element)
- list-block (zero or more)
- retrieve-marker (one or more, subject to constraints specified for this element)
- table (zero or more)
- table-and-caption (zero or more)
- wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- list-item

Attributes

The following attributes can be used on this element:

id	index-class
index-key	keep-together

For an example showing the use of the element see .

25.5.4 list-item-label

Description

This element contains the label part of a list item.

The list-item-label contains block-level elements, it does not itself contain text.

The list-item-label should always have its end-indent attribute set to "label-end()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the end-indent is not so specified the label column will overlap the body column.

The list-item-body should always have its start-indent attribute set to "body-start()" which is a function returning a value calculated from the provisional-distance-between-starts and provisional-label-separation attributes. If the start-indent is not so specified the label column will overlap the body column.

Child element(s)

This element can contain the following elements:

- block (zero or more)
- block-container (zero or more)
- float (one or more, cannot be used inside an out-of-line element)
- index-range-begin (one or more, subject to constraints specified for this element)
- index-range-end (one or more, subject to constraints specified for this element)
- list-block (zero or more)
- retrieve-marker (one or more, subject to constraints specified for this element)
- table (zero or more)
- table-and-caption (zero or more)
- wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- list-item

Attributes

The following attributes can be used on this element:

id	index-class
index-key	keep-together

For an example showing the use of the element see .

25.6 Dynamic effects: link and multi formatting objects

The objects described in this section are used to create links and dynamic content. As most of the elements in this section relate to dynamic content, which is not applicable to PDF, only basic-link is implemented.

25.6.1 basic-link

Description

This element is used to create a link in the PDF document, either to an external URL or to another location in the document.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

block (zero or more)

block-container (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

list-block (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- block
- inline
- folio-suffix
- folio-prefix
- bidi-override
- marker

Attributes

The following attributes can be used on this element:

- background-attachment
- background-color
- background-image
- background-repeat
- background-position-horizontal
- background-position-vertical
- border
- border-before-color
- border-before-style
- border-before-width
- border-after-color
- border-after-style
- border-after-width
- border-start-color
- border-start-style
- border-start-width
- border-end-color
- border-end-style
- border-end-width
- border-top
- border-top-color
- border-top-style
- border-top-width
- border-bottom
- border-bottom-color
- border-bottom-style
- border-bottom-width
- border-left
- border-left-color
- border-left-style

border-left-width	border-right
border-right-color	border-right-style
border-right-width	padding
padding-before	padding-after
padding-start	padding-end
padding-top	padding-bottom
padding-left	padding-right
relative-position	bottom
top	right
left	alignment-adjust
alignment-baseline	baseline-shift
destination-placement-offset	dominant-baseline
external-destination	id
index-class	index-key
internal-destination	keep-together
keep-with-next	keep-with-previous
line-height	show-destination
target-processing-context	target-presentation-context
target-stylesheet	

25.7 Formatting objects for bookmarks

The objects described in this section are used to create bookmarks which link to parts of the document and appear in a tree structure.

25.7.1 bookmark-tree

Description

This is the top level element in a tree of bookmarks, used to create the bookmark entries displayed on the right side in a PDF viewer.

Child element(s)

This element can contain the following elements:

bookmark (zero or more)

Parent element(s)

This element can be contained in the following elements:

root

Attributes

The following attributes can be used on this element:

starting-state

For an example showing the use of the element see Figure 25-18.

Figure 25-18: A bookmark tree

```
<bookmark-tree>
  <bookmark internal-destination="section-1">
    <bookmark-title>Chapter 1</bookmark-title>
    <bookmark internal-destination="section-1-1">
      <bookmark-title>Section 1</bookmark-title>
    </bookmark>
    <bookmark internal-destination="section-1-2">
      <bookmark-title>Section 2</bookmark-title>
    </bookmark>
  </bookmark>
  <bookmark internal-destination="section-2">
    <bookmark-title>Chapter 2</bookmark-title>
    <bookmark internal-destination="section-2-1">
      <bookmark-title>Section 1</bookmark-title>
    </bookmark>
  </bookmark>
</bookmark-tree>
```

25.7.2 bookmark

Description

This creates a single bookmark which links to a place in the PDF file. The destination in the PDF file is created by giving some formatting object an id attribute, then setting the internal-destination attribute on the bookmark to the value specified in the destination id.

Child element(s)

This element can contain the following elements:

bookmark (zero or more)

bookmark-title (exactly one)

Parent element(s)

This element can be contained in the following elements:

bookmark-tree

bookmark

Attributes

The following attributes can be used on this element:

internal-destination starting-state

For an example showing the use of the element see Figure 25-18.

25.7.3 bookmark-title

Description

This element holds the text for a bookmark entry.

Parent element(s)

This element can be contained in the following elements:

bookmark

For an example showing the use of the element see Figure 25-18.

25.8 Out-of-line formatting objects

The objects described in this section are used to create floats and footnotes.

25.8.1 float

Description

This element is used to position content either (a) at the top of a page or (b) to the side of a page so that text flows around it.

Child element(s)

This element can contain the following elements:

- block (zero or more)
- block-container (zero or more)
- float (one or more, cannot be used inside an out-of-line element)
- index-range-begin (one or more, subject to constraints specified for this element)
- index-range-end (one or more, subject to constraints specified for this element)
- list-block (zero or more)
- retrieve-marker (one or more, subject to constraints specified for this element)
- table (zero or more)
- table-and-caption (zero or more)
- wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- float
- footnote-body
- static-content
- table-caption
- block
- block-container

inline

folio-suffix

folio-prefix

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

float	clear
id	index-class
index-key	

For an example showing the use of the element see Figure 25-19.

Figure 25-19: The float element

```
<block>
  <float float="start">
   <block-container inline-progression-dimension="3cm" padding="5mm">
      <block padding="2mm" space-before.conditionality="retain" border="1pt solid
white" width="2cm">
         <block padding-left="2mm">
          <external-graphic src="url(ibexorange.jpg)" content-width='50%'/>
         </block>
     </block>
   </block-container>
 </float>
 <block padding-top="2mm" padding-bottom='2mm' space-before="9pt" font="10pt 'minion
regular'">
 This text should appear to the right of the image until we pass the bottom of the
image
 and then appear below the image as well.
 </block>
 <block font="10pt 'minion regular'">
 We have lots of text here just to show that it will be formatted in the correct way
and eventually
 there will be enough text to go past the image and appear below it on the page.
Then we will have some
 XML which shows how to acheive this effect.
 We have lots of text here just to show that it will be formatted in the correct way
and eventually
 there will be enough text to go past the image and appear below it on the page.
Then we will have some
 XML which shows how to acheive this effect.
 We have lots of text here just to show that it will be formatted in the correct way
and eventually
 there will be enough text to go past the image and appear below it on the page.
Then we will have some
 XML which shows how to acheive this effect.

 </block>

</block>
```

25.8.2 footnote

Description

This element is used to insert a footnote which will appear at the bottom of the region. The footnote contains an inline which is the *anchor* and is position in the containing block at the point the footnote occurs. The contents of the footnote-body are move out of line to the end of the region.

Child element(s)

This element can contain the following elements:

footnote-body (exactly one)

inline (exactly one)

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

block

inline

folio-suffix

folio-prefix

bidi-override

marker

Attributes

The following attributes can be used on this element:

id

index-class

index-key

25.8.3 footnote-body

Description

This element is used to insert a the content of a footnote which will appear at the bottom of the region.

Child element(s)

This element can contain the following elements:

block (zero or more)

block-container (zero or more)

float (one or more, cannot be used inside an out-of-line element)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

list-block (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

footnote

Attributes

The following attributes can be used on this element:

id index-class

index-key

25.9 Formatting objects for indexing

These objects are used in creating an index at the end of a document.

25.9.1 index-page-number-prefix

Description

This element is used to specify a prefix for page numbers created using an index-key-reference element.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

index-key-reference

25.9.2 index-page-number-suffix

Description

This element is used to specify a suffix for page numbers created using an index-key-reference element.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

index-key-reference

25.9.3 index-range-begin

Description

This element is used to indicate the start of a range of content which has an associated index key. The index will typically contain the range of page numbers between an index-range-begin and an index-range-end. An index-range-begin/index-range-end pair match if the ref-id property of the index-range-end has the same value as the id property on the index-range-begin.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

float

footnote-body

static-content

table-caption

block

block-container

inline

folio-suffix

folio-prefix

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

id index-class

index-key

25.9.4 index-range-end

Description

This element is used to indicate the end of a range of content which has an associated index key. The index will typically contain the range of page numbers between an index-range-begin and an index-range-end. An index-range-begin/index-range-end pair match if the ref-id property of the index-range-end has the same value as the id property on the index-range-begin.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

float

footnote-body

static-content

table-caption

block

block-container

inline

folio-suffix

folio-prefix

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

ref-id

25.9.5 index-key-reference

Description

This element is used in the index creation process to insert a set of page numbers for all occurrences of the specified index-key.

The child index-page-number-prefix and index-page-number-suffix elements specify content which will appear before and after the page numbers. This is how you would create page numbers like [20].

Child element(s)

This element can contain the following elements:

index-page-number-prefix (zero or one)

index-page-number-suffix (zero or one)

Parent element(s)

This element can be contained in the following elements:

index-page-citation-list

Attributes

The following attributes can be used on this element:

page-number-treatment ref-index-key

25.9.6 index-page-citation-list

Description

This element is used in the index creation process to group set of page numbers in the index.

Child element(s)

This element can contain the following elements:

index-key-reference (one or more)

index-page-citation-list-separator (zero or one)

index-page-citation-range-separator (zero or one)

Parent element(s)

This element can be contained in the following elements:

Attributes

The following attributes can be used on this element:

merge-sequential-page-numbers

merge-ranges-across-index-key-references

merge-pages-across-index-key-references

25.10 Other formatting objects

25.10.1 change-bar-begin

Description

This element marks the start of a change region, and causes a change bar to be drawn on the side of the containing region from the point this element occurs to the location of the matching change-bar-end element.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

Attributes

The following attributes can be used on this element:

change-bar-class	change-bar-color
change-bar-offset	change-bar-placement
change-bar-style	z-index

25.10.2 change-bar-end

Description

This element marks the end of a change region, and causes a change bar to be drawn on the side of the containing region from the location of the matching change-bar-begin this element of this element.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

Attributes

The following attributes can be used on this element:

change-bar-class	change-bar-color
change-bar-offset	change-bar-placement
change-bar-style	z-index

25.10.3 wrapper

Description

This element is used to specify inherited attributes for the elements it contains.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

block (zero or more)

block-container (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

list-block (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

- index-page-citation-range-separator
- index-page-citation-list-separator
- index-page-number-prefix
- index-page-number-suffix
- wrapper
- basic-link
- title
- float
- footnote-body
- static-content
- table-caption
- block
- block-container
- inline
- folio-suffix
- folio-prefix
- inline-container
- bidi-override
- table-cell
- list-item-label
- list-item-body
- marker

Attributes

The following attributes can be used on this element:

- id
- index-class
- index-key

25.10.4 marker

Description

This element contains some content which will be retrieved elsewhere in the document using a retrieve-marker element.

Typically marker is used to set some piece of text such as the current chapter title which is then retrieved within a static-content element for placing in the page header. The Ibex manual uses this technique to place the current chapter name in the top right corner of most pages.

An marker cannot be used in static-content elements, and a retrieve-maker can be used only in static-content elements.

An marker uses the marker-class-name attribute to group markers which have a common purpose. The retrieve-marker element has some attributes to specify which marker should be retrieved, such as the first or last one in the document or the first or last one on that page.

Child element(s)

This element can contain the following elements:

text

basic-link (zero or more)

bidi-override (zero or more)

block (zero or more)

block-container (zero or more)

character (zero or more)

external-graphic (zero or more)

float (one or more, cannot be used inside an out-of-line element)

footnote (one or more, cannot be used inside an out-of-line element)

index-page-citation-last (zero or more)

index-range-begin (one or more, subject to constraints specified for this element)

index-range-end (one or more, subject to constraints specified for this element)

inline (zero or more)

inline-container (zero or more)

instream-foreign-object (zero or more)

leader (zero or more)

list-block (zero or more)

page-number (zero or more)

page-number-citation (zero or more)

page-number-citation-last (zero or more)

retrieve-marker (one or more, subject to constraints specified for this element)

scaling-value-citation (zero or more)

table (zero or more)

table-and-caption (zero or more)

wrapper (one or more, subject to constraints specified for this element)

Parent element(s)

This element can be contained in the following elements:

Attributes

The following attributes can be used on this element:

marker-class-name

For an example showing the use of the element see .

25.10.5 retrieve-marker

Description

The marker element contains some content which will be retrieved elsewhere in the document using a retrieve-marker element.

Typically marker is used to set some piece of text such as the current chapter title which is then retrieved within a static-content element for placing in the page header. The Ibex manual uses this technique to place the current chapter subject in the footer.

The marker element cannot be used in static-content elements and the retrieve-maker element can be used only in static-content elements.

An marker uses the marker-class-name attribute to group markers which have a common purpose. The retrieve-marker element has some attributes to specify which marker should be retrieved, such as the first or last one in the document or the first or last one on that page.

For the retrieve-marker element to work its retrieve-class-name attribute must have the same value as the maker-class-name attribute used on some marker element.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

index-page-citation-range-separator

index-page-citation-list-separator

index-page-number-prefix

index-page-number-suffix

wrapper

basic-link

title

float

footnote-body

static-content

table-caption

block

block-container

inline

folio-suffix

folio-prefix

inline-container

bidi-override

table-cell

list-item-label

list-item-body

marker

Attributes

The following attributes can be used on this element:

retrieve-class-name

retrieve-position

retrieve-boundary

For an example showing the use of the element see .

25.10.6 retrieve-table-marker

Description

The marker element contains some content which will be retrieved elsewhere in the document using a retrieve-table-marker or retrieve-marker element.

The retrieve-table-marker element is used inside a table-header or table-footer to specify a marker whose content will be retrieved. This is described in detail on page 91

For the retrieve-marker element to work its retrieve-class-name attribute must have the same value as the maker-class-name attribute used on some marker element.

Child element(s)

This element can contain the following elements:

This element must be empty.

Parent element(s)

This element can be contained in the following elements:

Attributes

The following attributes can be used on this element:

retrieve-class-name retrieve-position-within-table

retrieve-boundary-within-table

For an example showing the use of the element see .

25.11 Attributes

25.11.1 absolute-position

Default value

auto

Values

auto
absolute
fixed
inherit

25.11.2 alignment-adjust

Description

This is used on a formatting objects to help explicitly determine the baseline for objects such as images which do not have a baseline.

Default value

auto

25.11.3 alignment-baseline

Description

This is used to specify which baseline an object should be aligned on. See page 61 for a discussion of baselines.

Default value

auto

Values

inherit
auto
baseline

before-edge
text-before-edge
central
middle
after-edge
text-after-edge
ideographic
alphabetic
hanging
mathematical

25.11.4 allowed-height-scale

Description

Sets possible scaling values for images. Not used in PDF creation.

Default value

any

25.11.5 allowed-width-scale

Description

Sets possible scaling values for images. Not used in PDF creation.

Default value

any

25.11.6 background-attachment

Description

Specifies whether background images scroll nor not. Not used in PDF creation.

Default value

any

25.11.7 background-color

Description

Sets the background color for the element.

Default value

transparent

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
transparent	
inherit	

25.11.8 background-image

Description

Specifies a URL for an image to be displayed in the background of an element. See page 95 for an example.

Default value

none

25.11.9 background-position-horizontal

Description

Specifies the initial horizontal position of a background image. See page 95 for an example.

Default value

0%

25.11.10 background-position-vertical

Description

Specifies the initial vertical position of a background image. See page 95 for an example.

Default value

0%

25.11.11 background-repeat

Description

Specifies if a background image should be repeated when it is smaller than the containing area. See page 95 for an example.

Default value

repeat

Values

repeat	The image is repeated horizontally and vertically
repeat-x	The image is repeated horizontally only
repeat-y	The image is repeated vertically only
no-repeat	The image is not repeated

25.11.12 baseline-shift

Description

Specifies the amount by which text in an inline should be shifted from the baseline. This is used to create subscript and superscript text. See page 62 for an example.

Default value

baseline

Values

baseline	Text is not shifted
sub	Text is lowered by an amount read from the font file

super	Text is raised by an amount read from the font file
percentage	Text is raised or lowered by a percentage of the current font size
length	Text is raised or lowered by the specified amount

25.11.13 blank-or-not-blank

Description

This is used on a repeatable-page-master-alternative to specify whether the page master should be selected when the current page is blank. The current page will be blank if an extra page is being generated in a page sequence because it must have an odd or even number of pages, or because the following page sequence must start on an odd or even page.

Default value

any

Values

any	
blank	
not-blank	Text is raised by an amount read from the font file

25.11.14 block-progression-dimension

Description

Sets the dimension of content in the block progression direction, which for a non-rotated page is down the page.

The content of an element excludes padding and borders. This means an element with block-progression-dimension="3cm" and border=".25cm" will have a height including borders and padding of 3.5cm.

Can be set as a single value such as:

Figure 25-20:

```
block-progression-dimension="20cm"
```

or you can specify minimum and maximum values like this:

Figure 25-21:
```
block-progression-dimension.minimum="5cm"
block-progression-dimension.maximum="25cm"
```

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
<percentage>	A percentage such as "10%". The value is calculated as a percentage of the parent elements height.
<length-range>	The value has three sub-components, namely minimim, optimum and maximum. Each of these can be set to a <length> value.
inherit	

25.11.15 border

Description

Sets the border for all four sides of an element to the same value.

Any of the values listed can be combined, for example you can have:

Figure 25-22:
```
border="12pt solid red"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
<border-width>	Can be any of the values: thin A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.

	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.16 border-after-color

Description

Sets the "after" border color, which for a non-rotated object is the bottom one. For example:

Figure 25-23:

```
border-after-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.17 border-after-style

Description

Sets the "after" border style, which for a non-rotated object is the bottom one. For example:

Figure 25-24: `border-after-style="solid"`

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example

		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.18 border-after-width

Description

Sets the "after" border width, which for a non-rotated object is the bottom one. For example:

Figure 25-25:

```
border-after-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)

inherit

25.11.19 border-before-color

Description

Sets the "before" border color, which for a non-rotated object is the top one. For example:

Figure 25-26: `border-before-color="red"`

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.20 border-before-style

Description

Sets the "before" border style, which for a non-rotated object is the top one. For example:

Figure 25-27: `border-before-style="solid"`

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset

	outset	See example this is 2pt blue outset
	groove	See example this is 2pt blue groove
	ridge	See example this is 2pt blue ridge

inherit

25.11.21 border-before-width

Description

Sets the "before" border width, which for a non-rotated object is the top one. For example:

Figure 25-28:

```
border-before-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)

inherit

25.11.22 border-bottom

Description

Sets the color, width and style of the bottom border of an element.

A shorthand way of setting border-bottom-color, border-bottom-width and border-bottom-style.

Any of the values listed can be combined, for example you can have:

Figure 25-29:

```
border-bottom="12pt solid red"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
<border-width>	Can be any of the values: thin — A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically. medium — A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically. thick — A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically. <length> — A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values: none — No border solid — A single solid line this is 1pt solid black double — Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double dashed — See example this is 2pt black dashed dotted — See example

	this is 2pt black dotted
inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
	this is 2pt blue inset
outset	See example
	this is 2pt blue outset
groove	See example
	this is 2pt blue groove
ridge	See example
	this is 2pt blue ridge

inherit

25.11.23 border-bottom-color

Description

Sets the bottom border color. For example:

Figure 25-30:

```
border-bottom-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.24 border-bottom-style

Description

Sets the bottom border style. For example:

Figure 25-31:

```
border-bottom-style="solid"
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.25 border-bottom-width

Description

Sets the bottom border width. For example:

Figure 25-32:

```
border-bottom-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values: thin — A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically. medium — A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically. thick — A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically. <length> — A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit	

25.11.26 border-collapse

Description

Controls whether borders on adjacent rows, cells and table elements are collapsed into a single border or remain separate.

Default value

collapse

Values

collapse	borders are collapsed. Precedence rules are evaluated to see which borders take precedence.
collapse-with-precedence	borders are collapsed. Precedence rules are evaluated to see which borders take precedence. In addition the border-precedence attribute can be used to change the precedence rules.
separate	borders are not collapsed. Only cell and table borders are considered, borders on all other elements are ignored.
inherit	

25.11.27 border-color

Description

Sets the border color for all four sides of an element to the same color or to a number of different colors.

To set all borders to the same color use a single value like this:

Figure 25-33:
```
border-color="red"
```

If there are two values the top and bottom borders are set to the first value and the side borders are set to the second, like this:

Figure 25-34:
```
border-color="red blue"
```

If there are three values the top border is set to the first value, the side borders are set to the second, and the bottom is set to the third like this:

Figure 25-35:
```
border-color="red blue green"
```

If there are four values the top border is set to the first value, the right border is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

Figure 25-36:
```
border-color="red blue green black"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
transparent	
inherit	

25.11.28 border-end-color

Description

Sets the end border color (the right side of a non-rotated page). For example:

Figure 25-37:
```
border-end-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.29 border-end-style

Description

Sets the end border style (the right side of a non-rotated page). For example:

Figure 25-38:
```
border-end-style="solid"
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example

		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.30 border-end-width

Description

Sets the end border width (the right side of a non-rotated page). For example:

Figure 25-39:

```
border-end-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)

inherit

25.11.31 border-left

Description

Sets the color, width and style of the left border of an element.

A shorthand way of setting border-left-color, border-left-width and border-left-style.

Any of the values listed can be combined, for example you can have:

Figure 25-40:

```
border-left="12pt solid red"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.
<border-width>	Can be any of the values: thin — A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically. medium — A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically. thick — A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically. <length> — A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values: none — No border solid — A single solid line this is 1pt solid black double — Two lines separated by a gap. The gap is 1/3 of the width of the border. this is 4pt black double dashed — See example this is 2pt black dashed dotted — See example this is 2pt black dotted

	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.32 border-left-color

Description

Sets the left border color. For example:

Figure 25-41:

```
border-left-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.33 border-left-style

Description

Sets the left border style. For example:

Figure 25-42:

```
border-left-style="solid"
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.34 border-left-width

Description

Sets the left border width. For example:

Figure 25-43:

```
border-left-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

25.11.35 border-right

Description

Sets the color, width and style of the right border of an element.

A shorthand way of setting border-right-color, border-right-width and border-right-style.

Any of the values listed can be combined, for example you can have:

Figure 25-44:
```
border-right="12pt solid red"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.	
<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.

	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.36 border-right-color

Description

Sets the right border color. For example:

Figure 25-45:

```
border-right-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.37 border-right-style

Description

Sets the right border style. For example:

Figure 25-46:

```
border-right-style="solid"
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.38 border-right-width

Description

Sets the right border width. For example:

Figure 25-47:
```
border-right-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
inherit		

25.11.39 border-separation

Description

Sets the separation between cell borders in a table with border-collapse="separate".

To set both horizontal and vertical separation the same use:

Figure 25-48:
```
border-separation="3pt"
```

To set horizontal and vertical separation to different values use the inline-progression-dimension and block-progression-dimension components like this:

Figure 25-49:
```
border-separation.inline-progression-dimension="3pt"
border-separation.block-progression-dimension="10pt"
```

For a non-rotated page block-progression-dimension is down the page and inline-progression-dimension is across.

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

inherit

25.11.40 border-spacing

Description

This is a shorthand method of setting the border-separation attribute.

To set both horizontal and vertical separation the same use:

Figure 25-50:
```
border-spacing="3pt"
```

To set horizontal and vertical separation to different values use two values separated by a space like this:

Figure 25-51:
```
border-spacing="3mm 13mm"
```

The first value sets the horizontal spacing, the second sets the vertical spacing.

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

inherit

25.11.41 border-start-color

Description

Sets the start border color (the left side of a non-rotated page). For example:

Figure 25-52:
```
border-start-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.42 border-start-style

Description

Sets the start border style (the left side of a non-rotated page). For example:

Figure 25-53:
```
border-start-style="solid"
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example

		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.43 border-start-width

Description

Sets the start border width (the left side of a non-rotated page). For example:

Figure 25-54:

```
border-start-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)

inherit

25.11.44 border-style

Description

Sets the border style for all four sides of an element to the same style or to a number of different styles.

To set all borders to the same style use a single value like this:

Figure 25-55:
```
border-style="solid"
```

If there are two values the top and bottom borders are set to the first value and the side borders are set to the second, like this:

Figure 25-56:
```
border-style="solid none"
```

If there are three values the top border is set to the first value, the side borders are set to the second, and the bottom is set to the third like this:

Figure 25-57:
```
border-style="solid none double"
```

If there are four values the top border is set to the first value, the right border is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

Figure 25-58:
```
border-style="solid none double dotted"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example

		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge
transparent		
inherit		

25.11.45 border-top

Description

Sets the color, width and style of the top border of an element.

A shorthand way of setting border-top-color, border-top-width and border-top-style.

Any of the values listed can be combined, for example you can have:

Figure 25-59:

```
border-top="12pt solid red"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.	
<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.

	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example
		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.46 border-top-color

Description

Sets the top border color. For example:

Figure 25-60:

```
border-top-color="red"
```

Default value

the value of the color property

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

inherit

25.11.47 border-top-style

Description

Sets the top border style. For example:

Figure 25-61:

```
border-top-style="solid"
```

Default value

none

Values

<border-style>	Can be any of the values:	
	none	No border
	solid	A single solid line
		this is 1pt solid black
	double	Two lines separated by a gap. The gap is 1/3 of the width of the border.
		this is 4pt black double
	dashed	See example
		this is 2pt black dashed
	dotted	See example
		this is 2pt black dotted
	inset	The top and left borders are slightly darker then the required color and the bottom and right borders are slightly lighter.
		this is 2pt blue inset
	outset	See example
		this is 2pt blue outset
	groove	See example

		this is 2pt blue groove
	ridge	See example
		this is 2pt blue ridge

inherit

25.11.48 border-top-width

Description

Sets the top border width. For example:

Figure 25-62:

```
border-top-width="1pt"
```

Default value

medium

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)

inherit

25.11.49 border-width

Description

Sets the border width for all four sides of an element to the same width or to a number of different widths.

To set all borders to the same width use a single value like this:

Figure 25-63:
```
border-width="1pt"
```

If there are two values the top and bottom borders are set to the first value and the side borders are set to the second, like this:

Figure 25-64:
```
border-width="1pt 3pt"
```

If there are three values the top border is set to the first value, the side borders are set to the second, and the bottom is set to the third like this:

Figure 25-65:
```
border-width="1pt 2pt 3pt"
```

If there are four values the top border is set to the first value, the right border is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

Figure 25-66:
```
border-width="1pt 2pt 3pt 4pt"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

<border-width>	Can be any of the values:	
	thin	A thin border. The actual default width is Settings.BorderWidthThin and so can be changed programatically.
	medium	A medium border. The actual default width is Settings.BorderWidthMedium and so can be changed programatically.
	thick	A thick border. The actual default width is Settings.BorderWidthThick and so can be changed programatically.
	<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points)
transparent		
inherit		

25.11.50 bottom

Description

This is used for absolutely and relatively positioned elements only. It sets the distance from the bottom edge of the containing element to the bottom edge of this element.

Default value

auto

25.11.51 break-after

Description

Use this element to insert a page break after this element.

Default value

auto

Values

auto	
column	
page	A page break will occur after this element.
inherit	

25.11.52 break-before

Description

Use this element to insert a page break before this element.

Default value

auto

Values

auto	
column	
page	A page break will occur before this element.
inherit	

25.11.53 caption-side

Description

This is used on a table-and-caption to specify on which side of the table the caption appears.

Default value

before

Values

before
after
start
end
top
bottom
left
right
inherit

25.11.54 character

Description

This attribute sets the character to be inserted by a character element. For instance to insert the character "A" into the content you would use an character element like this:

Figure 25-67:

```
<character character="A"/>
```

Default value

This attribute has no default value, you must provide a value.

25.11.55 clear

Description

This is used on an element to specify that it may not be placed next to a float element on one or both sides.

Default value

none

Values

left
right
both
none
inherit

25.11.56 clip

Description

Not used in PDF creation.

Default value

auto

25.11.57 color

Description

This sets the foreground color of text.

Default value

inherited from parent

Values

<color>	A color such as 'red', 'blue' etc. or an RGB color such as '#445566' or a CMYK color defined using the rgb-icc color.

25.11.58 color-profile-name

Description

This is used in specifying color profiles for PDF/X documents. The value must be set to "cmyk". See page 143 for more information.

Default value

auto

25.11.59 column-count

Description

Sets the number of columns in a body region. Only the body region can have more than one column.

For example to create a body region with three columns set column-count to 3 like this:

Figure 25-68:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"
            background-color="#eeeeee" column-count="3"/>
      </simple-page-master>
   </layout-master-set>

   <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
</root>
```

Default value

1

Values

<integer>	A non-negative integer. Sets the number of columns to this value.

25.11.60 column-gap

Description

Sets the gap between columns in a body region with column-count > 1. Only the body region can have more than one column.

For example to create a body region with two columns separated by a 4cm gap set column-count to 2 and column-gap to "4cm" like this:

Figure 25-69:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"
            column-count="2" column-gap="4cm"/>
      </simple-page-master>
   </layout-master-set>

   <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>Hello World</block>
      </flow>
   </page-sequence>
</root>
```

Default value

12.0pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.61 column-number

Description

This is used on a table-column element to specify which column the table-column element refers to.

This attribute is optional as the column number can be determined from the position of the table-column element in the list of such elements.

Figure 25-70:

```
<table>
    <table-column column-number="1"
      column-width="20%"/>
    <table-column column-number="2"
      column-width="30%"/>
    <table-column column-number="3"
      column-width="50%"/>
    <table-body>
         <table-row>
              <table-cell>col 1</table-cell>
              <table-cell>col 2</table-cell>
              <table-cell>col 3</table-cell>
         </table-row>
    </table-body>
</table>
```

Default value

current column number

25.11.62 column-width

Description

This is used on a table-column element to specify the width of the column the table-column element refers to.

For example to set the widths of three columns to 20%, 30% and 50% you would do this:

Figure 25-71:

```
<table>
    <table-column column-number="1"
      column-width="20%"/>
    <table-column column-number="2"
      column-width="30%"/>
    <table-column column-number="3"
       column-width="50%"/>
    <table-body>
         <table-row>
              <table-cell>col 1</table-cell>
              <table-cell>col 2</table-cell>
              <table-cell>col 3</table-cell>
         </table-row>
    </table-body>
</table>
```

Default value

This attribute has no default value, you must provide a value.

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.63 content-height

Description

This is used on a graphic element such as a external-graphic to set the height of the image.

The size of an image and the size of the area containing it are two seperate things. The height and width attributes set the size of the area containing the image, the content-height and content-width attributes set the size of the image itself.

Percentage values refer to percentages of the actual size of the image as determined from the image file.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

scale-to-fit

25.11.64 content-width

Description

This is used on a graphic element such as a external-graphic to set the width of the image.

The size of an image and the size of the area containing it are two seperate things. The height and width attributes set the size of the area containing the image, the content-height and content-width attributes set the size of the image itself.

Percentage values refer to percentages of the actual size of the image as determined from the image file.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

scale-to-fit

25.11.65 display-align

Description

This attribute sets the vertical alignment of content contained within the element with this attribute.

Default value

inherited from parent

Values

auto	
before	Align to before edge which for non-rotated content is the top.
center	Align to center
after	Align to after edge which for non-rotated content is the bottom.

25.11.66 end-indent

Description

This attribute sets indentation of content from the end edge of the containing area. For non-rotated content the end edge is the right edge.

This attribute sets the indentation of the content contained in the element. The content will be positioned the required distance from the right edge of the containing area, and any padding and border will then be placed outside the content.

For CSS style alignment of nested elements use the margin-left and margin-right attributes instead of start-indent and end-indent.

Default value

opt

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

inherit

25.11.67 ends-row

Description

Within a table-body (or table-header and table-footer) element a table has table-cell elements. Normally cells are placed inside a table-row element, but it is possible to place the cells directly below the table-body element and not have any table-row elements. In this case the formatter determines formation of rows by looking for ends-row and

starts-row attributes on each table-cell. If a table-cell ends the row then the ends-row attribute should be set to "true", otherwise it should be set to "false" or not used at all.

A table which has two rows of three cells each and is created without row elements looks like this:

Figure 25-72:

```
<table>
    <table-body>
         <table-cell starts-row="true">col 1</table-cell>
         <table-cell>col 2</table-cell>
         <table-cell ends-row="true">col 3</table-cell>
         <table-cell starts-row="true">col 1</table-cell>
         <table-cell>col 2</table-cell>
         <table-cell ends-row="true">col 3</table-cell>
    </table-body>
</table>
```

Default value

false

Values

false	This cell does not end the row
true	This cell ends the row

25.11.68 extent

Description

The extent attribute determines how large a region is. It is used on region elements other then the region-body element.

The extent is the size of the region. The outer edge of the region is calculated from the edge of the page plus any margin on the simple-page-master element. The inner edge of the region is the outer edge plus the value of the extent attribute.

Percentage values refer to the size of the page.

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.69 external-destination

Description

This attribute destination of a basic-link element used to create a hyperlink in the document.

The format of the external-destination attribute must be a URI specification (RFC2396) as described below.

To link local file the format should be:

Figure 25-73:
```
external-destination="url(external.pdf)"
```

or to link to a website use a format like this:

Figure 25-74:
```
external-destination
    ="url(http://www.xmlpdf.com/builds/ibex.pdf)"
```

Default value

Values

<uri-specification>	A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

25.11.70 float

Description

Specifies how the block which is floated should be positioned. Specify float="start" or float="before" to move the block to the start of the page. Specify float="left" to position content to the left side of the page and have other content flow around the right side of the positioned content.

Default value

none

Values

inherit
before
start
end
left
right

25.11.71 flow-map-name

Description

Specifies a unique name for a flow-map. page-sequence elements which will be laid out with that flow-map will have the same value for their flow-map-reference attribute.

Default value

none, a value is required

25.11.72 flow-map-reference

Description

This is used on a page-sequence to specify the name of a flow-map which will control the layout of content (in flows) to regions.

It is also used on a flow-name-specifier to specify the name of a flow which will be mapped using the containing flow-map.

Default value

none, a value is required

25.11.73 flow-name

Description

This attribute is used on flow and static-content elements to define which region the content from the flow and static-content is placed.

For content to be placed on the page the flow-name attribute must correspond to the region-name attribute of a region of the current page layout. If the flow-name is not the

same as one of the region names the content contained in the flow and static-content is discarded.

Default value

This attribute has no default value, you must provide a value.

Values

<name>	use a value which matches a region-name used on one of the regions on the current simple-page-master.

25.11.74 font

Description

This attribute is shorthand for the font-style, font-variant, font-weight, font-size, font-family, line-height attributes.

Typically the font attribute will be set to a value which defines the font name and size plus possibly bold or italic. Some example of this are:

Figure 25-75:
```
font="12pt arial"
```

Figure 25-76:
```
font="bold 12pt arial"
```

Figure 25-77:
```
font="bold 12pt "minion regular""
```

The elements of the font attribute must be specified in this order:

style (normal, italic)

variant (normal, smallcaps)

weight (bold, bolder, lighter etc.)

size (1em, 12pt)

line height (12pt/14pt)

font-family (helvetica)

If the font name contains spaces it should be placed in quotes. If the attribute value is in single quotes, place the font name in double quotes like this:

Figure 25-78:
```
font="12pt "minion regular" "
```

If the attribute value is in double quotes, place the font name in single quotes like this:

Figure 25-79:
```
font="12pt "minion regular" "
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

25.11.75 font-family

Description

This sets the font family for the element.

This attribute can be set to a single font name like this:

Figure 25-80:
```
font-family="arial"
```

or a list of fonts separated by commas, like this:

Figure 25-81:
```
font-family="arial, "minion regular""
```

If the font name contains spaces it should be placed in quotes. If the attribute value is in single quotes, place the font name in double quotes like this:

Figure 25-82:
```
font="12pt "minion regular" "
```

If the attribute value is in double quotes, place the font name in single quotes like this:

Figure 25-83:
```
font="12pt "minion regular" "
```

In addition to actual font names the following values can be used:

"serif", "sans-serif", "cursive", "fantasy", "monospace"

These names are mapped to actual font names by the Settings.

Default value

The value of Settings.DefaultFontFamily or inherited from parent

25.11.76 font-size

Description

This sets the font size of this element and the elements it contains.

Typical values are show here:

Figure 25-84:
```
font-size="12pt"
```

Figure 25-85:
```
font-size="1.2em"
```

Values which set the font size relative to the font size of the containing element can also be used like this:

Figure 25-86:
```
font-size="smaller"
```

Percentage sizes refer to the font size of the containing element.

Default value

medium

Values

xx-small	Set the font size to the value of Settings.XX_Small which defaults to "7pt"
x-small	Set the font size to the value of Settings.X_Small which defaults to "8.3pt"
small	Set the font size to the value of Settings.Small which defaults to "10pt"
medium	Set the font size to the value of Settings.Medium which defaults to "12pt"
large	Set the font size to the value of Settings.Large which defaults to "14.4pt"
x-large	Set the font size to the value of Settings.X_Large which defaults to "17.4pt"
xx-large	Set the font size to the value of Settings.XX_Small which defaults to "20.7pt"
smaller	Set the font size to the parent font size multipled by the value of Settings.Smaller which defaults to "0.8em"

larger	Set the font size to the parent font size multipled by the value of Settings.Larger which defaults to "1.2em"
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.77 font-style

Description

This sets the font style of this element and elements it contains.

Typical values are show here:

Figure 25-87:
```
font-style="italic"
```

Default value

normal

Values

normal	The style is the same as the style of the parent element.
italic	The style is italic.
oblique	The style is italic.
inherit	

25.11.78 font-weight

Description

This sets the font weight of this element and elements it contains.

The specification supports numeric values. These are mapped as follows:

900	bold
800	bold
700	bold
600	normal
500	normal
400	normal
300	normal
200	normal
100	normal

Default value

normal

Values

100-900	See the table above
normal	The weight is inherited from the parent element.
bold	The text will be bold.
inherit	

25.11.79 format

Description

In conjuction with grouping-separator and grouping-size this attribute sets the format to be used when formatting page numbers contained within this page-sequence.

Default value

1

Values

1	Use numeric formatting so page numbers will be 1,2,3 ..
i	Use roman formatting so page numbers will be i, ii, iii, iv, v ..

25.11.80 height

Description

Sets the height of the content of an element excluding padding and borders. This means an element with height="3cm" and border=".25cm" will have a height including borders and padding of 3.5cm.

This example shows the effect of the height attribute on the content of the block-container:

Figure 25-88:

```
<block-container space-before="1cm" height="3cm"
         border=".5cm solid red">
         <block>3+.5</block>
 </block-container>
```

This produces this output:

By pressing Control-U in Acrobat Reader you can measure the content and see that the area *within the borders* is 3cm high.

Note that the width and height attributes do not apply to the fo:block element.

To set minumum and maximum height values use the block-progression-dimension attribute.

Default value

auto

25.11.81 id

Description

This attribute is set on elements which need to be referenced from somewhere else in the document.

An example of this is the page-number-citation element which inserts into the document the page number some other content appears on. The content whose page number we want to retrieve is given an id attribute, and the page-number-citation sets its ref-id attribute to the same value.

An example of this is:

Figure 25-89:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <layout-master-set>
      <simple-page-master master-name="simple">
         <region-body margin="2.5cm" region-name="body"/>
      </simple-page-master>
   </layout-master-set>

  <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block id="22">
         Hello
         </block>
      </flow>
  </page-sequence>

  <page-sequence master-reference="simple">
      <flow flow-name="body">
         <block>
           The block with id="22" is on page
           <page-number-citation ref-id="22"/>
         </block>
      </flow>
  </page-sequence>

</root>
```

Default value

a unique value generated by Ibex

Values

<id>	a unique string

25.11.82 initial-page-number

Description

This attribute sets the page number of the first page create by a page-sequence element.

Default value

auto

25.11.83 inline-progression-dimension

Description

Sets the dimension of content in the inline progression direction which for a non-rotated page is across the page.

The content of an element excludes padding and borders. This means an element with inline-progression-dimension="3cm" and border=".25cm" will have a width including borders and padding of 3.5cm.

Can be set as a single value such as:

Figure 25-90:
```
inline-progression-dimension="20cm"
```

or you can specify minimum and maximum values like this:

Figure 25-91:
```
inline-progression-dimension.minimum="5cm"
inline-progression-dimension.maximum="25cm"
```

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
<length-range>	The value has three sub-components, namely minimim, optimum and maximum. Each of these can be set to a <length> value.
inherit	

25.11.84 internal-destination

Description

This sets the destination of a basic-link element.

This should be set a value used as the id attribute of the element to be linked to.

Default value

''

25.11.85 keep-together

Description

Set this attribute to "always" to keep content together on one page. If content with keep-together="always" will not fit on what remains of a page it will be moved to the next page. If it is larger than the region in which it is being placed it will be split.

Default value

auto

25.11.86 keep-with-next

Description

Set this attribute to "always" to keep the content with the next element in the FO. If both elements do not fit on a page they will both be moved to the next page.

This is typically used to keep a heading together with the content which follows.

Any number of elements can be kept together by having keep-with-next="always" set on each one. If the list to be kept together exceeds the size of the region in which they are being placed they will not be kept together.

Default value

auto

25.11.87 keep-with-previous

Description

Set this attribute to "always" to keep the content with the previous element in the FO. If both elements do not fit on a page they will both be moved to the next page.

This is typically used to keep a the last two rows of a table together so that a single row is never displayed by itself.

Any number of elements can be kept together by having keep-with-previous="always" set on each one. If the list to be kept together exceeds the size of the region in which they are being placed they will not be kept together.

Default value

auto

25.11.88 leader-length

Description

This sets the length of a leader element.

This can be set as three components for the minimum, optimum and maximum values like this:

Figure 25-92:
```
leader-length.mimimum="10pt"
leader-length.optimum="50%"
leader-length.maximum="100%"
```

Or alternatively all three components can be set to the same value like this:

Figure 25-93:
```
leader-length="100%"
```

The default values for each component are:

leader-length.mimimum="0pt"

leader-length.optimum="12pt"

leader-length.maximum="100%"

Default value

see description

25.11.89 leader-pattern

Description

This sets the appearance of a leader element.

Default value

space

Values

space	The leader will be blank. This is useful for justification of text.
dots	The leader will be a dotted line.
rule	The leader will be a solid line.

25.11.90 left

Description

When used on an absolutely or relatively positioned element this sets the offset from the left edge of the container to the left edge of this element.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.91 linefeed-treatment

Description

This sets the way in which linefeeds in the FO appear in the output. By default linefeeds are treated as spaces only. See page 69 for a detailed example of the effects of this attribute.

Default value

treat-as-space

Values

treat-as-space	Linefeeds in the FO become spaces in the output.
preserve	Linefeeds in the FO become linefeeds in the output.

25.11.92 line-height

Description

Sets the height of a line.

Percentage values refer to the current font size.

Default value

normal

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
normal	Line height is 1.2 times the font size.

25.11.93 margin

Description

This is a shorthand way of setting margin-top, margin-bottom, margin-right and margin-left.

To set all margins to the same size use a single value like this:

Figure 25-94:
```
margin="1pt"
```

If there are two values the top and bottom margins are set to the first value and the side margins are set to the second, like this:

Figure 25-95:
```
margin="1pt 3pt"
```

If there are three values the top margin is set to the first value, the side margins are set to the second, and the bottom is set to the third like this:

Figure 25-96:
```
margin="1pt 2pt 3pt"
```

If there are four values the top margin is set to the first value, the right margin is set to the second, the bottom is set to the third and the left is set to the forth (so clockwise from the top) like this:

Figure 25-97:
```
margin="1pt 2pt 3pt 4pt"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.94 margin-bottom

Description

Sets the bottom margin on an element. For example:

Figure 25-98:
```
margin-bottom="1pt"
```

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

25.11.95 margin-left

Description

Sets the left margin on an element. For example:

Figure 25-99:
```
margin-left="1pt"
```

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

25.11.96 margin-right

Description

Sets the right margin on an element. For example:

Figure 25-100:
```
margin-right="1pt"
```

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

25.11.97 margin-top

Description

Sets the top margin on an element. For example:

Figure 25-101:
```
margin-top="1pt"
```

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

25.11.98 marker-class-name

Description

Sets the name for a marker which is then used on a retrieve-marker element to retrieve the content contained in that marker.

See marker for an example.

Default value

This attribute has no default value, you must provide a value.

25.11.99 master-name

Description

This is a unique name given to a simple-page-master or page-sequence-master and then used as the master-reference attribute of a page-sequence to specify which page master will be used to lay out the content of the page-sequence.

Default value

This attribute has no default value, you must provide a value.

25.11.100 master-reference

Description

Both simple-page-master and page-sequence-master have a unique master-name attribute which is used as the master-reference attribute of a page-sequence to specify which page master will be used to lay out the content of the page-sequence.

Default value

This attribute has no default value, you must provide a value.

25.11.101 number-columns-repeated

Description

This is used on a table-column element to indicate how many columns the table-column element applies to.

Default value

1

25.11.102 number-columns-spanned

Description

This is used on a table-cell element to specify how many columns the cell spans. This is functionally similar to the HTML colspan attribute.

Default value

1

25.11.103 number-rows-spanned

Description

This is used on a table-cell element to specify how many rows the cell spans. This is functionally similar to the HTML rowspan attribute.

Default value

1

25.11.104 orphans

Description

This specifies the number of lines of text which must appear in a paragraph before a page break. At the default setting of "2" a single line will never appear by itself at the bottom of a page. If there is room for only a single line on a page (and the paragraph has more than one line) the whole paragraph will be shifted to the next page.

Increasing the value increases the number of lines which must appear on a page before a page break.

See also widows.

Default value

2

25.11.105 overflow

Description

This attribute determines whether content which exceeds the size of an element should be displayed or not.

An example of this is the fixed size region elements such as region-before which have their size set by the extent attribute. If content is placed in the region using a static-content element the content may be too large for the region. If this happens and the overflow attribute is set to "hidden" the content will not appear.

Default value

auto

Values

hidden	Content which exceeds the elements boundaries will be discarded.
auto	Content which exceeds the elements boundaries will be displayed.
visible	Content which exceeds the elements boundaries will be displayed.

25.11.106 padding

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This is a shorthand way of setting padding-top, padding-bottom, padding-right and padding-left.

To set all paddings to the same size use a single value like this:

Figure 25-102:
```
padding="1pt"
```

Default value

Shorthand properties do not have default values. See individual properties for their default values.

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.107 padding-after

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the after edge of an element which for an non-rotated area is the bottom edge.

Figure 25-103: `padding-after="1pt"`

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.108 padding-before

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the before edge of an element which for an non-rotated area is the top edge.

Figure 25-104: `padding-before="1pt"`

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.109 padding-bottom

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the bottom edge of an element.

Figure 25-105:

```
padding-bottom="1pt"
```

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.110 padding-end

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the end edge of an element which for an non-rotated area is the right edge.

Figure 25-106:

```
padding-end="1pt"
```

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.111 padding-left

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the left edge of an element.

Figure 25-107:

```
padding-left="1pt"
```

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.112 padding-right

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the right edge of an element.

Figure 25-108:

```
padding-right="1pt"
```

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.113 padding-start

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the start edge of an element which for an non-rotated area is the left edge.

Figure 25-109:
```
padding-start="1pt"
```

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.114 padding-top

Description

Padding is space which appears between the border of an element and the content (such as text) of that element.

This sets the padding on the top edge of an element.

Figure 25-110:
```
padding-top="1pt"
```

Default value

opt

Values

inherit	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.115 page-height

Description

This is used on a simple-page-master element to set the height of a page.

If not set or if set to "auto" the page height is determined from the Settings.PageHeight property.

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.116 page-width

Description

This is used on a simple-page-master element to set the width of a page.

If not set or if set to "auto" the page width is determined from the Settings.PageWidth property.

Default value

auto

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.117 precedence

Description

This is used on region-before and region-after elements to control whether the top and bottom regions take precedence over (i.e. extend into the corners over) the side regions.

Default value

false

Values

false

true

25.11.118 provisional-distance-between-starts

Description

This applies to the list-block element and sets the distance (in each list-item) between the start of the label element and the start of the body element.

This is not the same as the width of the label element because the width of the label element is reduced by the provisional-label-separation value.

See list-block for an example.

Default value

24pt

25.11.119 provisional-label-separation

Description

This applies to the list-block element and sets the distance between the end of the label element and the start of the body element.

See list-block for an example and also provisional-distance-between-starts.

Default value

6pt

25.11.120 reference-orientation

Description

This attribute is used to set the rotation of whole pages (when used on simple-page-master), regions (when used on region element), blocks (when used on block-container) and table elements.

Rotation is counter-clockwise.

See block-container for an example.

Default value

0

Values

0
90
180
270
-90
-180
-270
inherit

25.11.121 ref-id

Description

This attribute is used on the page-number-citation to identify which the element for which we want to retrieve the page number. This should match the value of the id attribute on the other element.

See page-number-citation for an example.

Default value

This attribute has no default value, you must provide a value.

25.11.122 region-name-reference

Description

This is used on a region-name-specifier to specify the name of a region which will have flows mapped onto it by the containing flow-map.

Default value

none, a value is required

25.11.123 retrieve-boundary

Description

This attribute is used on a retrieve-marker to specify limits on which markers should be retrieved.

See marker for a complete example.

Default value

page-sequence

Values

page
page-sequence
document

25.11.124 retrieve-class-name

Description

This attribute is used on a retrieve-marker to specify which marker is to be retrieved. This attribute specifies which class of marker is retrieved and the retrieve-boundary and retrieve-position attributes are used to choose one of the markers in that class.

See marker for a complete example.

Default value

This attribute has no default value, you must provide a value.

25.11.125 retrieve-position

Description

This attribute is used on a retrieve-marker to specify which marker is to be retrieved. The retrieve-class-name attribute specifies which class of marker is retrieved and the retrieve-boundary and retrieve-position attributes are used to choose one of the markers in that class.

See marker for a complete example.

Default value

first-starting-within-page

Values

first-starting-within-page	Use the first marker which appears starts on this page
first-including-carryover	Use the first marker which has any content on this page
last-starting-within-page	
last-ending-within-page	

25.11.126 right

Description

For an absolutely positioned element this specifies the distance between the right edge of the containing element and the right edge of this element.

Default value

auto

25.11.127 rule-thickness

Description

This is used on the leader element to specify the thickness (i.e. height) of the line the leader creates.

Default value

1pt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.128 scaling

Description

This is used on graphic elements external-graphic and instream-foreign-object to specify how the image should be scaled.

If the scaling is uniform a change to the image size using content-height or content-width will result in a corresponding change in the other dimension to preserve the aspect ratio. If scaling is non-uniform a change to height or width will not change the other dimension and the aspect ratio will be changed.

Default value

uniform

Values

uniform	See above.
non-uniform	See above.

25.11.129 space-after

Description

This attribute is used to define the amount of space which appears between this element and the next.

This attribute can be set as a single value like this:

Figure 25-111:

```
space-after="3mm"
```

or individual components can be set like this:

Figure 25-112:

```
space-after.minimum="3pt"
space-after.optimum="4pt"
space-after.maximum="5pt"
```

Space resolution in XSL-FO is complicated. If two elements have space after the first one and before the second one, usually the space is combined using a formula so that generally speaking the largest space will be used.

For example if there are two blocks A and B, and A has space-after="3cm" and B has space-before="2cm" the space between the blocks will not be the sum of the two spaces (ie. 5cm) it will be the largest of the two, ie. 3cm.

To prevent the two spaces from being merged, and get the sum of the two spaces you can use the precedence component like this:

Figure 25-113:

```
space-after="3cm" space-after.precedence="force"
```

Precedence can also be assigned a number. If there are two spaces to be merged and they have different precedence values the one with the highest value will be used. For example:

Figure 25-114:

```
<block space-after="3cm" space-after.precedence="5">
A
</block>
<block space-before="1cm" space-after.precedence="6">
B
</block>
```

In this case the space between the two blocks will be 1cm because the second block has the higher precedence value so its space value is the one which is used.

Space which appears before a block at the top of a region is usually discarded. To avoid this and make the space appear use the conditionality component like this:

Figure 25-115:

```
space-before="3cm" space-before.conditionality="retain"
```

To make matters even more complex, the space after an element refers to the space between the last mark made by this element and the first mark made by the next element. This means we need to consider child elements of the two elements whose space is being merged.

For example the block A below has a child block A2 which has a space-after attribute. This means when Ibex merges the space between A and B, it also considers the space between A2 and B.

Figure 25-116:

```
<block space-after="3cm" >
   A
<block space-after="4cm" >
  A2
</block>
</block>
<block space-before="1cm" >
  B
</block>
```

so the space between A and B will be 4cm because this is the largest value. If B had a child block this would also be considered.

And it gets worse. In the example shown above A2 makes the last mark on the page made by the A block and its children. If A had a bottom border, this border would then be the last mark made by the A block and its children (because the border of A is after

A2) and the merging formula would not consider A2 (as it does not now make the last mark) and so the gap between A and B would now be 3cm.

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.130 space-before

Description

This attribute is used to define the amount of space which appears between this element and the previous one.

This attribute can be set as a single value like this:

Figure 25-117:
```
space-before="3mm"
```

or individual components can be set like this:

Figure 25-118:
```
space-before.minimum="3pt"
space-before.optimum="4pt"
space-before.maximum="5pt"
```

Space resolution in XSL-FO is complicated. See space-after for a detailed description of space resolution.

Default value

opt

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.131 span

Description

This attribute is used on block-level (block,table,list-block) elements whose immediate parent element is a flow. The span indicates if the block element should span all the columns of a multi-column region-body. The only options are "none" and "all". It is not possible to span some but not all columns.

Default value

none

Values

none	Span one column
all	Span all columns
inherit	

25.11.132 src

Description

This specifies the source for a external-graphic element.

The src attribute is called a *uri-specification* and must follow the following rules:

> A sequence of characters that is "url(", followed by optional white space, followed by an optional single quote (') or double quote (") character, followed by a URI reference as defined in [RFC2396], followed by an optional single quote (') or double quote (") character, followed by optional white space, followed by ")". The two quote characters must be the same and must both be present or both be absent. If the URI reference contains a single quote, the two quote characters must be present and be double quotes.

This means the following are all valid values for the src attribute:

> uri(ibex.jpg)
>
> uri("ibex.jpg")
>
> uri('ibex.jpg')
>
> url(http://www.xmlpdf.com/images/download2.gif)

Default value

This attribute has no default value, you must provide a value.

25.11.133 start-indent

Description

This attribute sets indentation of content from the start edge of the containing area. For non-rotated content the start edge is the left edge.

This attribute sets the indentation of the content contained in the element. The content will be positioned the required distance from the right edge of the containing area, and any padding and border will then be placed outside the content.

For CSS style alignment of nested elements use the margin-left and margin-right attributes instead of start-indent and end-indent.

Default value

opt

Values

auto	
<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).
inherit	

25.11.134 starting-state

Description

Specifies if a bookmark entry should be open (with its child bookmarks visible) when the PDF file is opened.

Default value

show

Values

inherit
show
hide

25.11.135 starts-row

Description

Within a table-body (or table-header and table-footer) element a table has table-cell elements. Normally cells are placed inside a table-row element, but it is possible to place the cells directly below the table-body element and not have any table-row elements. In this case the formatter determines formation of rows by looking for ends-row and starts-row attributes on each table-cell. If a table-cell ends the row then the ends-row attribute should be set to "true", otherwise it should be set to "false" or not used at all.

A table which has two rows of three cells each and is created without row elements looks like this:

Figure 25-119:

```
<table>
 <table-body>
  <table-cell starts-row="true">col 1</table-cell>
   <table-cell>col 2</table-cell>
    <table-cell ends-row="true">col 3</table-cell>
    <table-cell starts-row="true">col 1</table-cell>
    <table-cell>col 2</table-cell>
    <table-cell ends-row="true">col 3</table-cell>
  </table-body>
</table>
```

Default value

false

Values

false	This cell does not start a new row.
true	This cell starts a new the row.

25.11.136 table-layout

Description

This attribute controls whether the layout of a table (which means the column widths) is calculated from the content of the cells or from table-column elements.

Use "fixed" to calculate column widths from table-column elements. **This is the recommended approach. It is faster and makes the output file look consistent when using different data.**

Default value

auto

Values

auto	
fixed	see above

25.11.137 table-omit-footer-at-break

Description

By default a table-footer element is repeated before every table break. If you set this attribute to "true" the table footer will appear only once, at the end of the table.

Default value

false

Values

true	Footer appears once at end of table
false	Footer appears at every page break

25.11.138 table-omit-header-at-break

Description

By default a table-header element is repeated after every table break. If you set this attribute to "true" the table header will appear only once, at the beginning of the table.

Default value

false

Values

true	header appears once at start of table
false	header appears after every page break

25.11.139 text-align

Description

This sets the text alignment of text contained within the element. This does not align the last (or only) line of a paragraph - see the text-align-last attribute for aligning the last line.

Default value

start

Values

start	same as left for non-rotated content
left	
center	
end	same as right for non-rotated content
right	
justify	

25.11.140 text-align-last

Description

This sets the text alignment of text last line of a paragraph - see the text-align attribute for aligning lines other than the last one.

Default value

start

Values

start	same as left for non-rotated content
left	
center	
end	same as right for non-rotated content
right	
justify	

25.11.141 white-space-collapse

Description

This controls how multiple contiguous whitespace in the FO is treated by Ibex. By default after processing of linefeeds all remaining runs of two or more consecutive spaces are replaced by a single space, then any remaining space immediately adjacent to a remaining linefeed is also discarded.

See page 69 for a detailed example of the effects of this attribute.

Default value

true

25.11.142 white-space-treatment

Description

This controls how whitespace characters in the FO are treated by Ibex.

See page 69 for a detailed example of the effects of this attribute.

Default value

ignore-if-surrounding-linefeed

25.11.143 widows

Description

This specifies the number of lines of text which must appear in a paragraph at the top of a page. At the default setting of "2" a single line will never appear by itself at the top of a page. If possible a line from the previous page will be moved to the the current page so that 2 lines of text appear at the top of the page. If this is not possible (perhaps because of the orphans setting) the whole paragraph will be moved to the current page.

Increasing the value increases the number of lines which must appear on a page.

See also orphans.

Default value

2

25.11.144 width

Description

This sets the desired width for an element. This is shorthand way of setting all three components of the inline-progression-dimension attribute.

Default value

auto

Values

<length>	A length such as '10cm'. Valid units are pt (points) cm (centimetres) in (inches) mm (millimetres) em (current font size in points).

25.11.145 wrap-option

Description

This option controls word wrapping within an element.

Default behavior for text within a block is for words which do not fit on one line to wrap to the next line and the height of the block to increase. If wrap-option="no-wrap" then words which do not fit on the first line are discarded if overflow="hidden".

See also overflow

Default value

wrap

Values

inherit	
wrap	words wrap to the next line
no-wrap	words do not wrap to the next line

25.11.146 z-index

Description

This attribute controls the positioning of one element over another. By default a more deeply nested element will appear over its container element but by changing the z-order you can change which elemnets appear over which other elements.

Default value

auto

Chapter 26

Compliance with the XSL-FO standard

This chapter describes which extent to which Ibex supports the XSL W3C Recommendation 1.1 dated 5 December 2006.

Objects are listed grouped together in the same way in which they appear in the specification.

26.1 Declarations and pagination and layout formatting objects

Element	Status
root	implemented
declarations	implemented
color-profile	implemented
page-sequence	implemented
page-sequence-wrapper	implemented
layout-master-set	implemented
page-sequence-master	implemented
single-page-master-reference	implemented
repeatable-page-master-reference	implemented
repeatable-page-master-alternatives	implemented
conditional-page-master-reference	implemented
layout-master-set	implemented
simple-page-master	implemented
region-body	implemented
region-before	implemented
region-after	implemented
region-start	implemented
region-end	implemented

Element	Status
flow	implemented
static-content	implemented
title	implemented
flow-map	implemented
flow-assignment	implemented
flow-source-list	implemented
flow-name-specifier	implemented
flow-target-list	implemented
region-name-specifier	implemented

26.2 Block-level formatting objects

Element	Status
block	implemented
block-container	implemented

26.3 Inline-level formatting objects

Element	Status
bidi-override	implemented
character	implemented
initial-property-set	not implemented
external-graphic	implemented
instream-foreign-object	implemented in .NET, not in Java
inline	implemented
inline-container	implemented
leader	implemented
page-number	implemented
page-number-citation	implemented
page-number-citation-last	implemented
folio-prefix	implemented
folio-suffix	implemented
scaling-value-citation	implemented

26.4 Formatting objects for tables

Element	Status
table-and-caption	implemented
table	implemented
table-column	implemented
table-caption	implemented
table-header	implemented
table-footer	implemented
table-body	implemented
table-row	implemented
table-cell	implemented

26.5 Formatting objects for lists

Element	Status
list-block	implemented
list-item	implemented
list-item-table-body	implemented
list-item-label	implemented

26.6 Dynamic effects: link and multi formatting objects

Element	Status
basic-link	implemented
multi-switch	not implemented, not relevant to PDF
multi-case	not implemented, not relevant to PDF
multi-toggle	not implemented, not relevant to PDF
multi-properties	not implemented, not relevant to PDF
multi-property-set	not implemented, not relevant to PDF

26.7 Formatting objects for indexing

Element	Status
index-page-number-prefix	implemented

Element	Status
index-page-number-suffix	implemented
index-range-begin	implemented
index-range-end	implemented
index-key-reference	implemented
index-page-citation-list	implemented
index-page-citation-list-separator	implemented
index-page-citation-range-separator	implemented

26.8 Formatting objects for bookmarks

Element	Status
bookmark-tree	implemented
bookmark	implemented
bookmark-title	implemented

26.9 Out-of-line formatting objects

Element	Status
float	implemented
footnote	implemented
footnote-body	implemented

26.10 Other formatting objects

Element	Status
change-bar-begin	implemented
change-bar-end	implemented
wrapper	implemented
marker	implemented
retrieve-marker	implemented
retrieve-table-marker	implemented